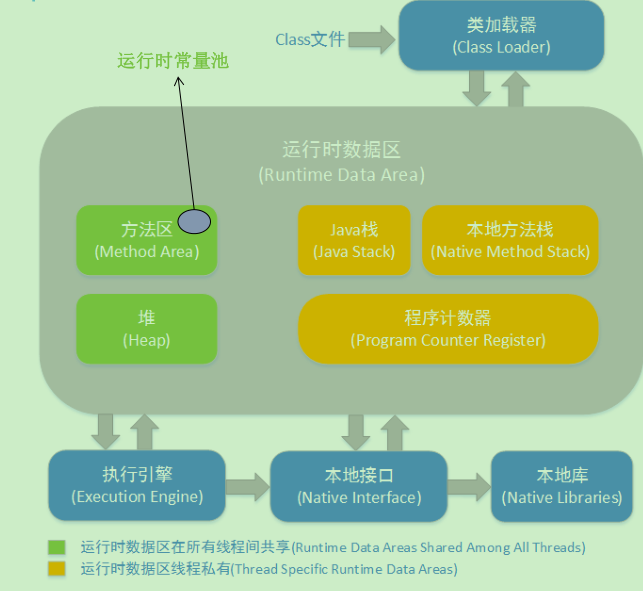
Jvm内存模型



程序计数器：较小的内存空间，

当前线程执行的字节码的行号指示器；各线程之间独立存储，互不影响；

java 栈：线程私有，生命周期和线程，每个方法在执行的同时都会创建一个 栈帧用于

存储局部变量表，操作数栈，动态链接，方法出口等信息。方法的执行就对应着栈帧在虚拟机栈中入栈和出栈的过程；栈里面存放着各种基本数据类型和对象的引用；(简单来说就是方法放在里面)

本地方法栈：本地方法栈保存的是native方法的信息，当一个JVM创建的线程调用native方法后，JVM不再为其在虚拟机栈中创建栈帧，

JVM只是简单地动态链接并直接调用native方法；

(简单来说就是放本地(c++/第三方)方法的,比如sdk包)

堆：Java堆是程序员需要重点关注的一块区域，因为涉及到内存的分配(new关键字，反射等)与回收(回收算法，收集器等)；(对象,字符串常量池)

方法区：也叫永久区，用于存储已经被虚拟机加载的类信息，常量("zdy","123"等)，静态变量(static变量)等数据。

(jdk1.8已经将方法区去掉了，将方法区移动到直接内存)(元空间,类放在里面)

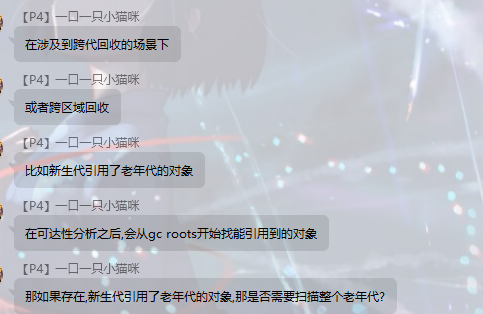
运行时常量池：

运行时常量池是方法区的一部分，用于存放编译期生成的各种字面("zdy","123"等)和符号引用。(元空间一部分,基本类型常量池)

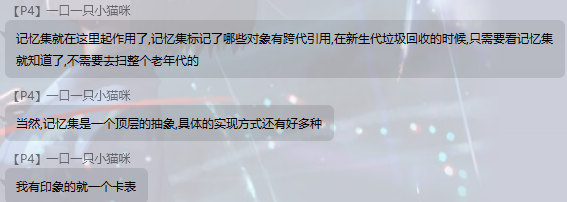
直接内存：不是虚拟机运行时数据区的一部分，也不是java虚拟机规范中定义的内存区域；

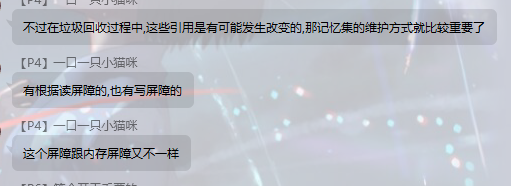
Gc回收算法

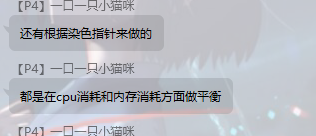
记忆集在gc回收中的作用

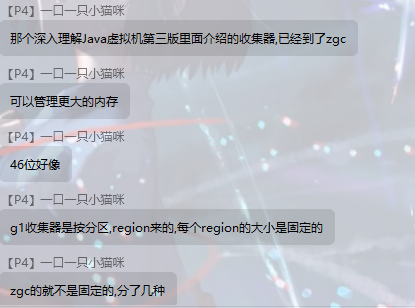












线程池

所有线程池都实现了这个类

ThreadPoolExecutor(int corePoolSize 核心线程数,  
 int maximumPoolSize 最大线程数,  
 long keepAliveTime 最大存活时间,  
 TimeUnit unit 时间单位,  
 BlockingQueue<Runnable> workQueue 队列, 队列,  
 ThreadFactory threadFactory 线程工厂,  
 RejectedExecutionHandler handler 拒绝策略)

ScheduledThreadPoolExecutor(int corePoolSize) {  
 super( corePoolSize, //核心线程数,就是你传的参数

Integer.*MAX\_VALUE* , // *最大线程数*(2^31)-1

0 , //存活时间

*NANOSECONDS* ,//时间单位*纳秒*  
 new DelayedWorkQueue()); //优先级队列