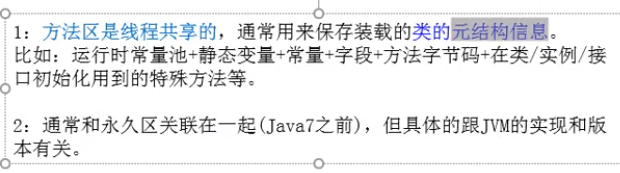
类加载器 自上往下使用加载器加载 找到就加载 找不到向下找 再找不到报错

BootStrapClassLoader 首先用这个加载 加载jdk提供的类

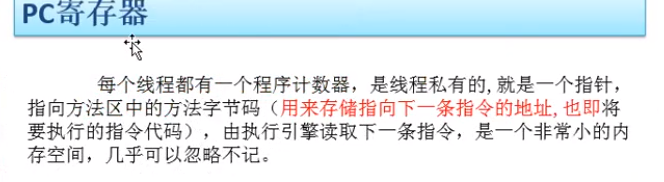
ExtClassLoader 加载jdk提供的类 加载jre\lib\ext里面的jar包

AppClassLoader 加载自己写的类 加载我们自己写的类

方法区:保存类的元结构



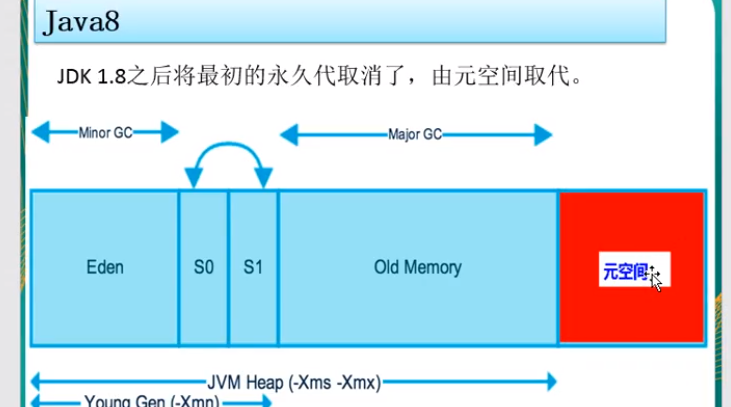
pc寄存器

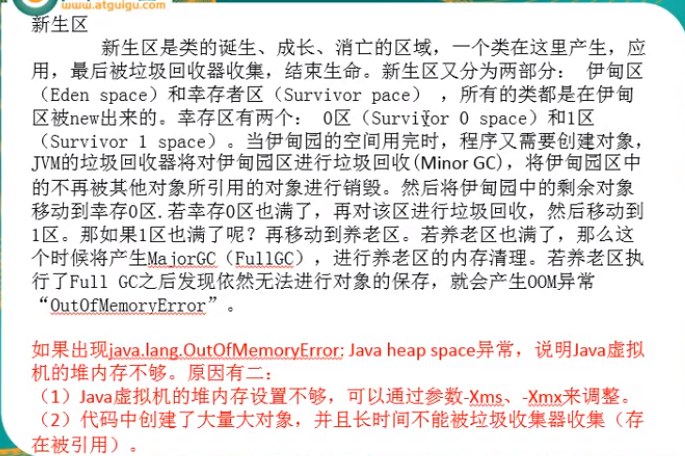


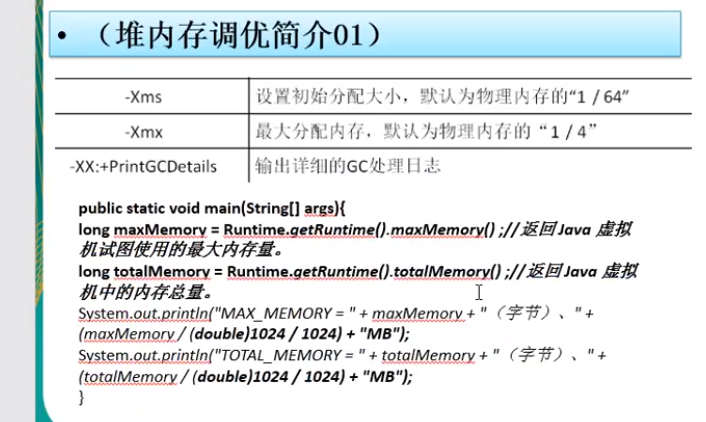
java7的时候堆的结构



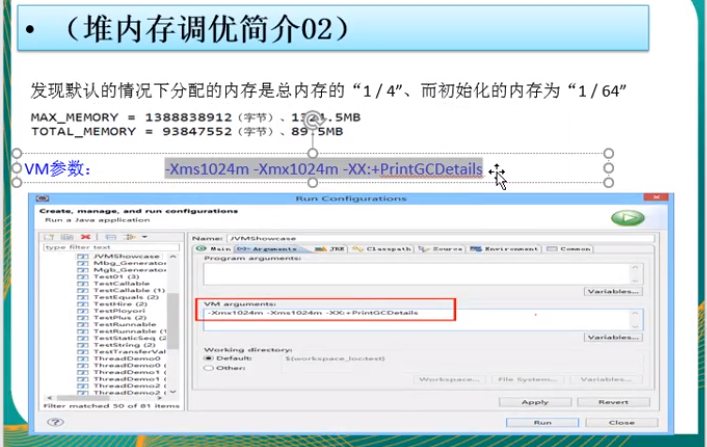
java8之后



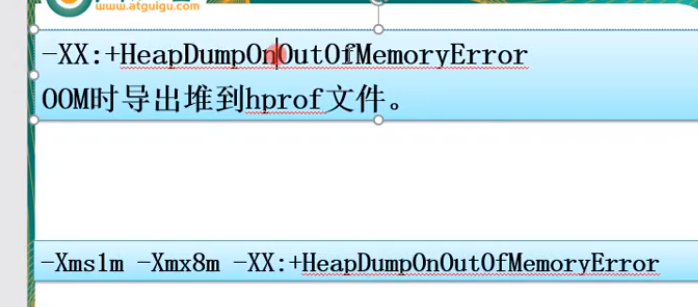




自定义初始化内存与最大内存



mat分析



通常我们定义一个基本数据类型的变量，一个对象的引用，还有就是函数调用的现场保存

都使用 JVM 中的栈空间

而通过 new 关键字和构造器创建的对象则放在堆空间，堆是垃圾收集

器管理的主要区域，由于现在的垃圾收集器都采用分代收集算法，所以堆空间还可以细分为新生代和老生代，再具体一点可以分为 Eden、Survivor（又可分为 From Survivor 和 To Survivor）、Tenured；方法区和堆都是各个线程共享的内存区域，用于存储已经被 JVM 加载的类信息、常量、静态变量、JIT 编译器编译后的代码等数据；程序中的字面量（literal）如直接书写的 100、"hello"和常量都是放在常量池中，常量池是方法区的一部分，。栈空间操作起来最快但是栈很小，通常大量的对象都是放在堆空间，栈和堆的大小都可以通过 JVM 的启动参数来进行调整，栈空间用光了会引发 StackOverflowError，而堆和常量池空间不足则会引发 OutOfMemoryError。

* -Xms / -Xmx — 堆的初始大小 / 堆的最大大小

• -Xmn — 堆中年轻代的大小

• -XX:-DisableExplicitGC — 让 System.gc()不产生任何作用

• -XX:+PrintGCDetails — 打印 GC 的细节

• -XX:+PrintGCDateStamps — 打印 GC 操作的时间戳

• -XX:NewSize / XX:MaxNewSize — 设置新生代大小/新生代最大大小

• -XX:NewRatio — 可以设置老生代和新生代的比例

• -XX:PrintTenuringDistribution — 设置每次新生代 GC 后输出幸存者乐园中对象年龄的分

布

• -XX:InitialTenuringThreshold / -XX:MaxTenuringThreshold：设置老年代阀值的初始值和最

大值

• -XX:TargetSurvivorRatio：设置幸存区的目标使用率