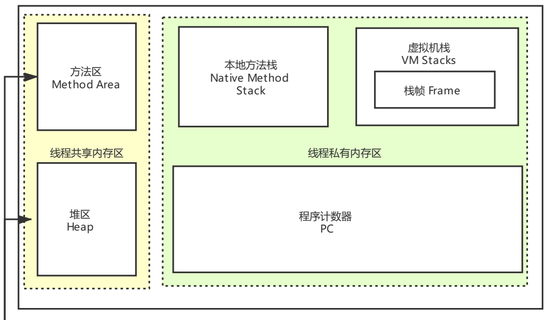
JVM内存五大区域



堆区（Heap）：用于存放所有对象

栈区（Stack）：用于存放基本数据类型的数据和对象的引用

方法区（Method Area）：用于存放类信息，常量，静态变量和编译后的字节码等等。（也称非堆，None-Heap）

程序计数器（PC）：用来指示当前线程所执行的字节码执行到第几行。可以理解为存储下一条指令的地址

GC垃圾回收器主要针对的是堆区，其次是方法区

用new创建的对象在堆区，函数中的临时变量在栈区，Java中的字符串在方法区（常量池）
Java方法执行内存模型，用于存储局部变量，操作数栈，动态链接，方法出口等信息，是线程隔离的

类加载器（Class Loader）

启动类加载器（Bootstrap ClassLoader）：最顶层的类加载器。负责加载JAVA_HOME\lib目录中的或-Xbootclasspath参数指定路径中的，且被虚拟机认可（按文件名识别，如rt.jar）的类

扩展类加载器（Extension ClassLoader）：负责加载JAVA_HOME\lib\ext目录中的，或通过java.ext.dirs系统变量指定路径中的类库

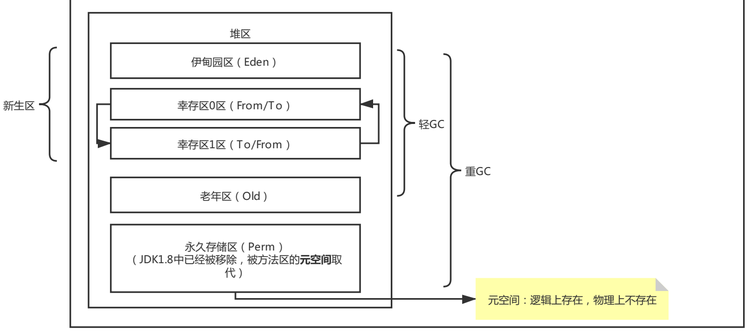
应用程序类加载器（Application ClassLoader）：也叫系统类加载器，可以通过getSystemClassLoader（）获取，负责加载用户路径（classpath）上的类库。如果没有自定义加载器，一般这个就是类加载器（默认）。

双亲委派模型

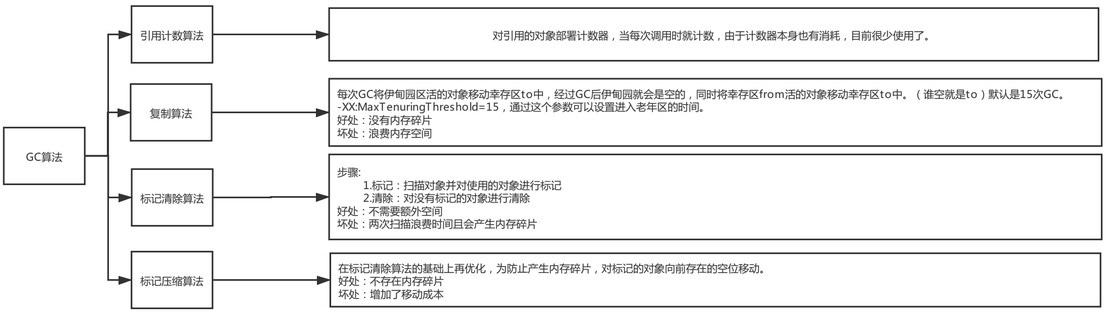
如果一个类接收到类加载请求，他自己不会去加载这个请求，而是将类加载请求委派给父类加载器，这样一层层向上传递，直到到达启动类加载器（Bootstrap ClassLoader）。只有当父类加载器无法加载这个请求时，子加载器才会尝试自己去加载。

作用：
1、防止重复加载同一个.class。通过委托去向上面问一问，加载过了，就不用再加载一遍。保证数据安全。
2、保证核心.class不能被篡改。通过委托方式，不会去篡改核心.class，即使篡改也不会去加载，即使加载也不会是同一个.class对象了。不同的加载器加载同一个.class也不是同一个Class对象。这样保证了Class执行安全。

堆区（Heap）：
一个JVM只有一个堆内存，堆内存的大小是可以调节的。
类加载器读取了类文件后，一般会把类，方法，常量，变量放到堆中，保存我们所有引用类型的真实对象；



OOM故障的处理：
1.利用内存快照分析工具：MAT, Jprofiler等等
2.DeBug：一行一行分析代码（实际上并不可取）



总结：
内存效率：复制算法>标记清除算法>标记压缩算法
内存利用率：标记压缩算法=标记清除算法>复制算法
内存整齐度：复制算法=标记压缩算法>标记清除算法

GC采用分代收集算法：对新生区采用复制算法，对老年区采用标记清除算法+标记压缩算法混合实现。