链滴

走进 JVM 之内存布局

作者：matthewhan

原文链接：https://ld246.com/article/1601263416585

来源网站：链滴

走进JVM之内存布局

cafe babe

内存布局

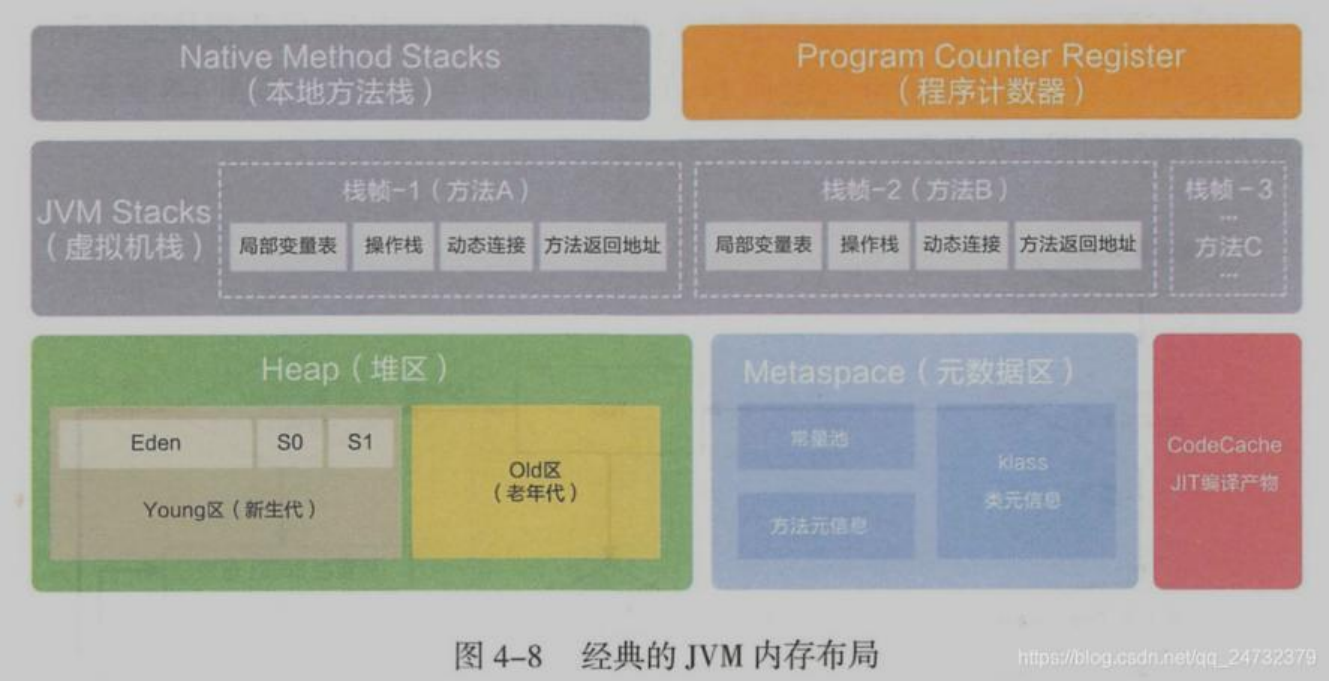

图 4-8　经典的 JVM 内存布局

1.堆（Heap）

存储着所有实例对象，GC的相关详见上一篇

2.元空间（Metaspace）

永久代和元空间都是对方法区的一个实现，方法区是一块所有线程共享的内存区域。Java7及之前为久代（Perm），之后均为元空间，所以类元信息、字段、静态属性、方法、常量等都移动至元空间。

区别于永久代，元空间在本地内存中分配。

3.虚拟机栈（JVM Stack）

JVM中的虚拟机栈是描述Java方法执行的内存区域，他是线程私有的。栈中的元素用于支持虚拟机方调用，每个方法从开始调用到执行完成的过程，就是栈帧从入栈到出栈的过程。

通过递归调用可以更好地理解方法的调用，尤其是二叉树的中序遍历，逐级返回有点类似JVM类加载制的双亲委派模型

局部变量表

存放方法参数和局部变量的区域

- 操作数栈
- 局部变量表
- 常量池引用

操作栈

操作站是一个初始状态为空的桶式结构栈。在方法执行的过程中，会有各种指令往栈中写入和提取信。

动态连接

每个栈帧包含一个常量池中对当前方法的引用，目的是支持方法调用过程的动态链接。

方法返回地址

1. 正常退出，一些关键字　return等
2. 异常退出

只要是退出都会返回到方法被调用的位置。方法退出的过程相当于弹出当前栈帧。

1. 返回值压入上层调用栈帧
2. 异常信息抛给能够处理的栈帧
3. PC计数器指向方法调用后的下一条指令

4.本地方法栈（Native Method Stacks）

线程对象私有，虚拟机栈主内，本地方法栈主外。

线程调用本地方法的时候，会进入一个不再受JVM约束的世界，优点：极高的执行效率、偏底层的跨程操作；缺点：威胁程序运行的稳定性。

JNI最著名的本地方法：System.currentTImeMillis()。

5.程序计数寄存器（Program Counter Register）

略

6.小结

从线程共享的角度来看，堆（Heap）和元空间（Metaspace）是线程共享的，其他都是线程内部私

的。

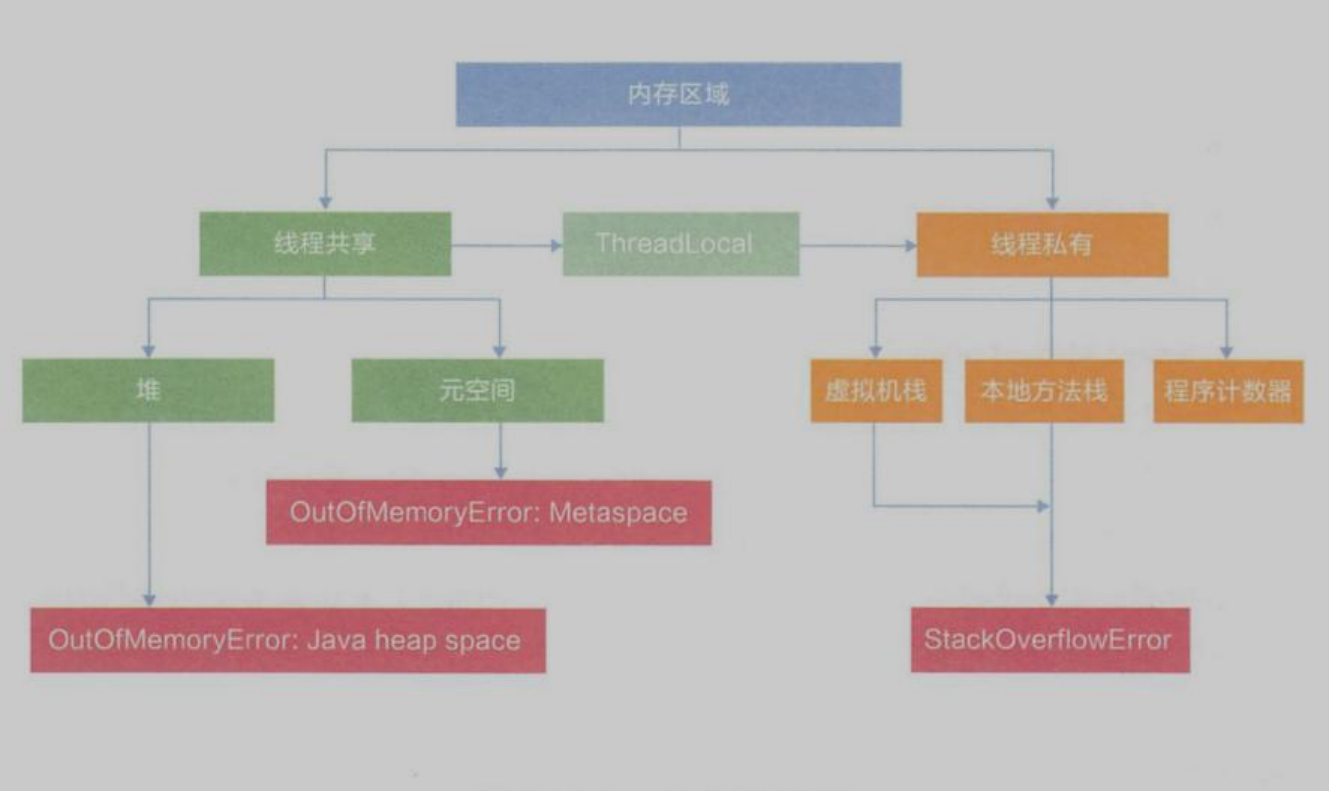

图 4-12 Java 的线程与内存