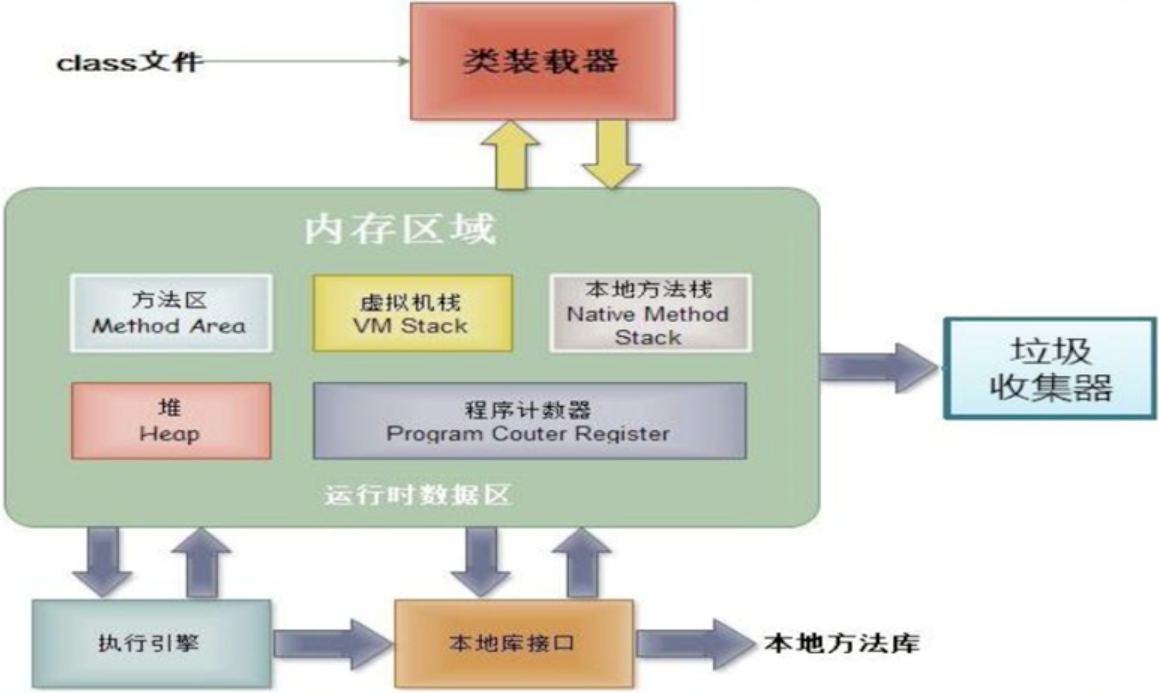
**JVM**



**1.Class Loader类加载器：**

**(只)负责加载class文件**

**2.Native Interface**

**本地库接口**： native 关键字 调用本地库接口, java native interface

**line 728： public void native** start0();

本地方法栈只存放本地方法库

**3.Method Area 方法区**：

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。 简单的说，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。

静态变量 + 常量 + 类信息 + 运行时常量池 存在方法区中 +

实例变量存在堆内存中。

**4.PC Register程序计数器**

每个线程都有一个程序计数器，就是一个指针，指向方法区中的方法字节码(下一个将要执行的指令代码)，由执行引擎读取下一条指令，是一个非常小的内存空间，几乎可以忽略不计。

**5.Native Method Stack(本地方法栈)**

Native Method Stack中登记native方法，在Execution Engine执行时加载native libraries。

**栈管运行， 堆管存储。**

**6.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结束，生命周期和线程一致，是线程私有的。基本类型的变量和对象的引用变量都是在函数的栈内存中分配。

**6.1. 栈存储是什么**

栈桢主要保存3种数据类型：

本地变量(Local Variables)：输入参数和输出参数以及方法内的变量；

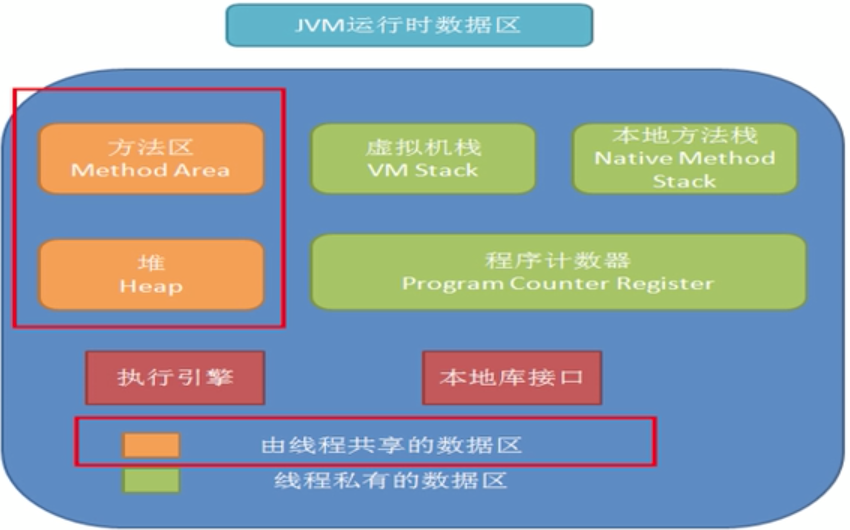
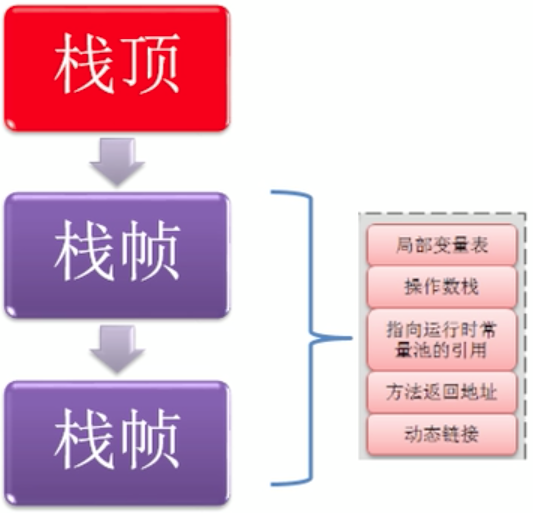
栈操作(Operand Stack):记录出栈、入栈的操作；

栈桢操作(Frame Data):包括类文件、方法等等。

**6.2 栈运行原理**

栈中的数据都是以栈桢的格式存在，栈桢是一个内存区块，是一个数据集。遵循“先进后出/后进先出”原则

顶部栈就是当前的方法，该方法执行完毕后会自动将此栈桢出栈。



**三种JVM JVM是一种规范 (起初7个)**

Sun公司的HotSpot、 BEA公司的JRockit 、I BM公司的J9 VM

**堆内存示意图**



**新生区:**

所有的类都是在伊甸区被new出来的。普通Minor GC和 Major GC(Full GC)

当Eden Space的空间用完时，JVM的垃圾回收器将对Eden Space进行垃圾回收(Minor GC),将Eden Space中不再被其他对象所引用的对象进行销毁。然后将Eden Space中剩余对象移动到幸存 0 区， 同理再到幸存1区，到养老区。

若养老区也满了，这个时候就会产生Major GC(Full GC)，进行养老区的内存清理。若养老区执行了Full GC后仍然无法进行对象的保存，就会产生OOM的异常“OutOfMemoryError”.

出现java.lang.OutOfMemoryError:Java Heap Space异常，说明Java虚拟机的堆内存不够，原因有2:

1. Java虚拟机的堆内存设置不够
2. 代码中创建了大量大对象，并且长时间不能被垃圾收集器收集(存在被引用)

**JVM大小默认理论值时出厂设置的1/4**

Runtime.*getRuntime*().maxMemory() / 1024 / 1024 单位时(MB)

**养老区:**

养老区用于保存从新生区筛选出来的JAVA对象，一般池对象都在这个区域活跃 (如数据库连接池)

**永久储存区:**

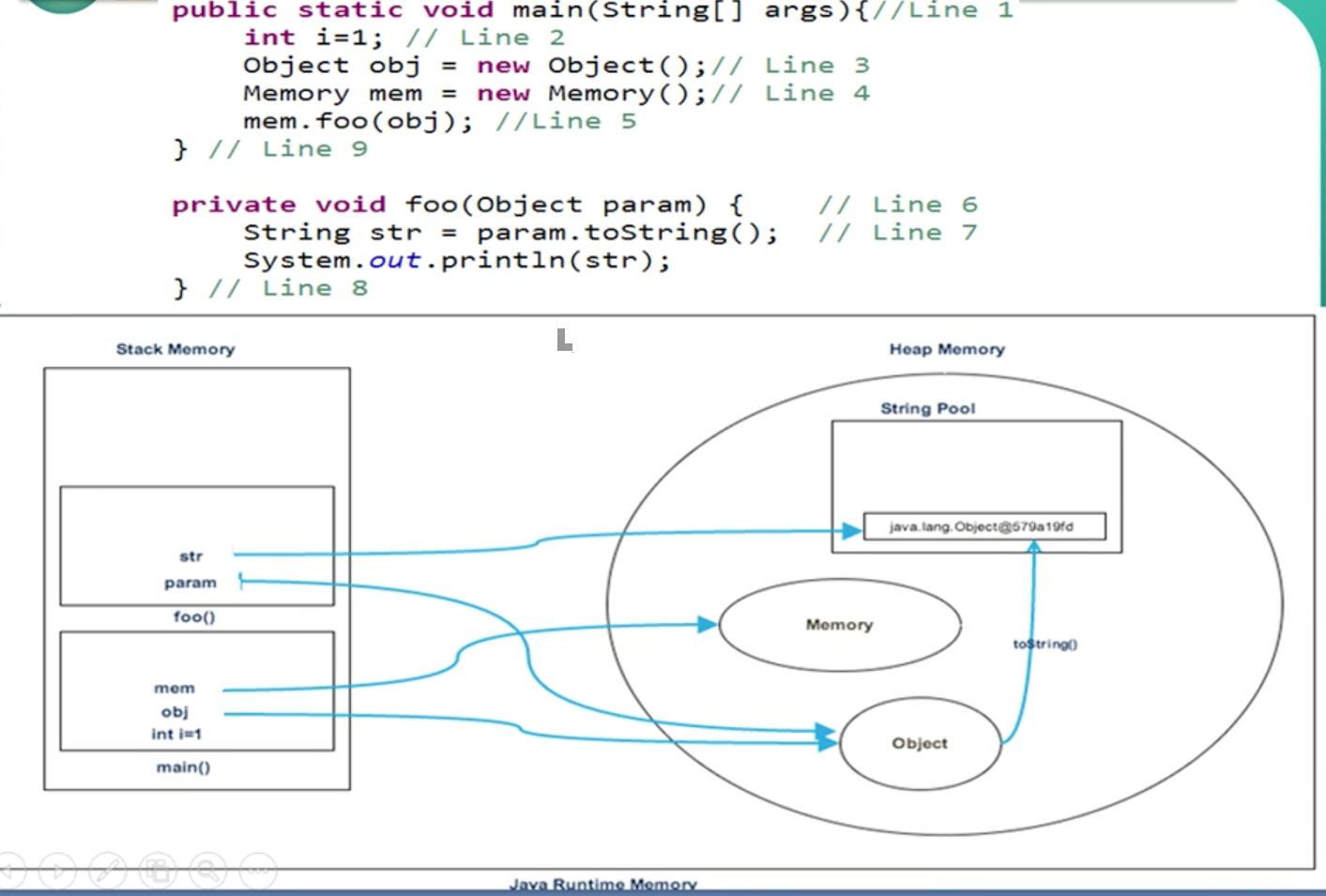
永久代没有垃圾回收这里指普通GC，永久存储区是一个常驻内存区域，用于存放JDK自身所携带的Class，Interface的元数据(如Object类),也就是说它存储的是运行环境必须的类信息，被装载进此区域的数据是不会被垃圾回收器回收掉，关闭JVM才会释放此区域所占用的内存。

Java 1.6及以前，有永久代，常量池1.6在方法区

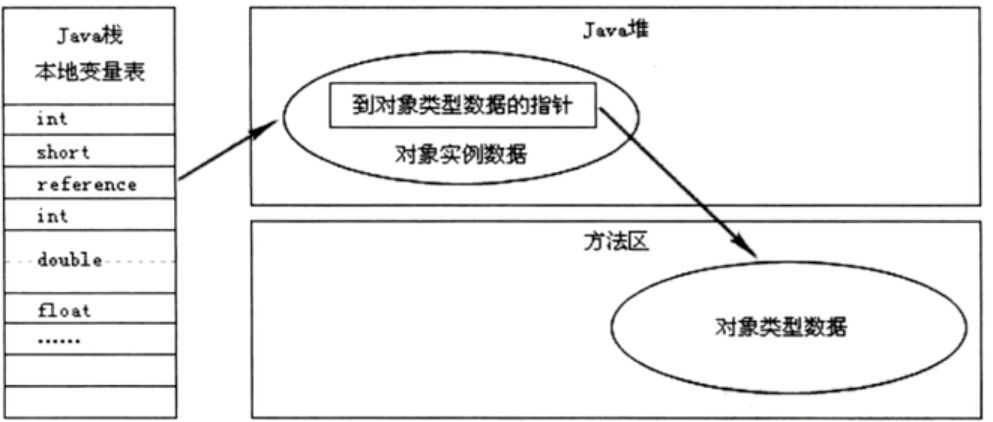
**Java 1.7: 有永久代**， 但逐步“去永久代”，常量池在 堆

**Java 1.8 无永久代，称为元空间** 常量池在元空间。

**栈管运行， 堆管存储。**

****

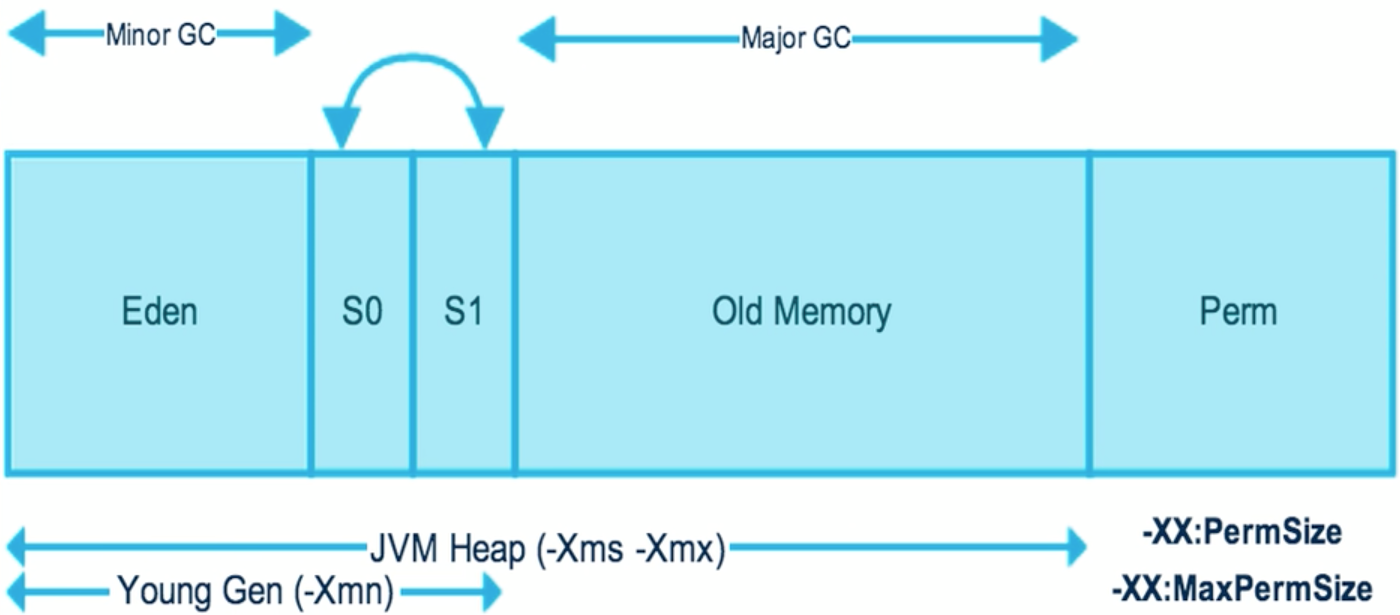
**严格来说： Heap和Method Area是分开的(1.6)**

****

逻辑上，堆由新生、养老、持久构成，实际上堆只有新生和养老构成。

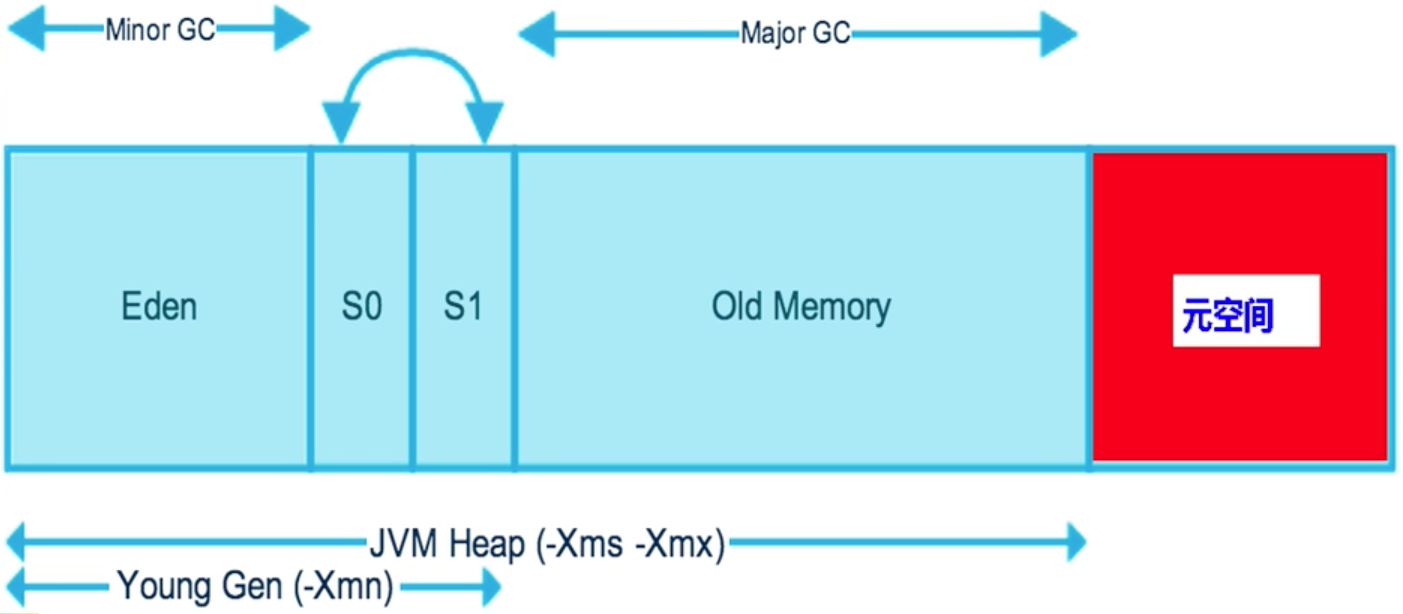


永久代是方法区的一个实现， jdk1.7的版本中，已经将原来放在永久代的字符串常量池移走。



Jdk1.7

1. Jdk 1.8之后将最初的永久代取消了，用**元空间**取代
2. 目的： 将HotSpot与JRockit两个虚拟机标准



Jdk 1.8

问：

1. 一般什么时候会发生GC？如何处理

答：Java中的GC会有2种回收：年轻代的Minor GC 和老年代的Full GC； 新对象创建时如果伊甸区空间不足会出发Minor GC，如果此时老年代的内存空间不足会触发Full GC，如果空间都不足抛出OutOfMemoryError。

1. GC回收策略

答：年轻代( 伊甸园区+2个幸存区)，GC的回收策略为“复制”

老年代的保存空间一般较大，GC回收策略为“整理-压缩”。

**年轻代**中使用的是Minor GC,这种GC算法采用的是 复制算法（Copying）复制算法没有内部碎片

**老年代**

标记清除(Mark-Sweep)

标记整理(Mark-Compact)

较优算法 🡺 **分代收集算法**