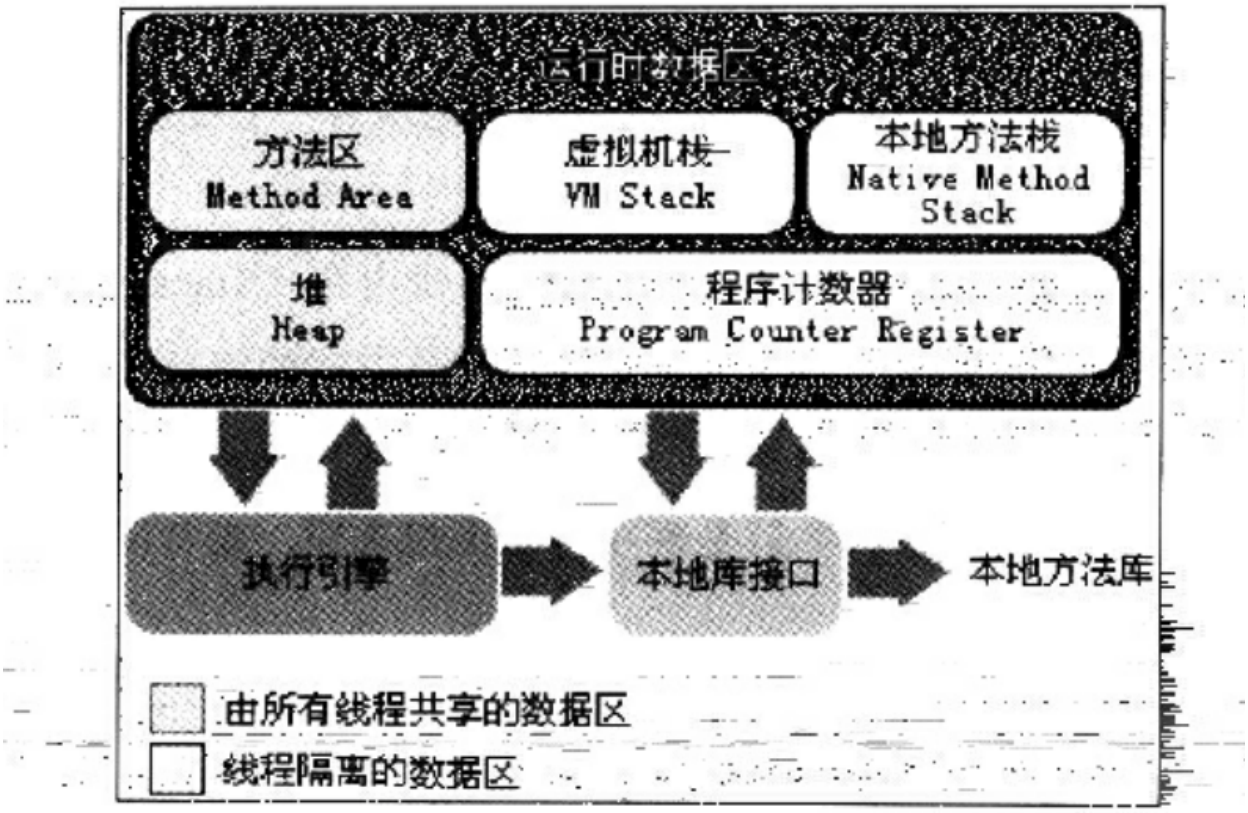

图 2-1　Java 虚拟机运行时数据区

一.程序计数器

1.当前线程执行字节码的行号指示器

2.每个线程有独立的计数器，为了线程切换时恢复到正确的位置

3.唯一一个java虚拟机规范中没有规定任何outOfMemoryError情况的区域

二.虚拟机栈

1.线程私有，生命周期与线程相同

2.描述了java方法执行的内存模型：每个方法在执行时都会创建一个栈帧用于存储局部变量表、操作数帧、动态链接、方法出口等信息。每一个方法调用直至完成的过程，就对应者一个帧栈在虚拟机栈中入栈到出栈的过程

3.其中局部变量表存放了编译期可知的各种基本数据类型、对象引用类型（可能指向对象起始地址的引用指针，也可能指向一个代表对象的句柄或一条字节码指令的地址），其中64位长度的long和double占用两个局部变量空间，其他类型只占一个，局部变量表所需的空间在编译期完成分配，当进入一个方法时，这个方法需要的帧中分配多大的局部变量空间是完全确定的，运行期间不会改变

4.在java虚拟机规范中，对这个区域规定了两种异常：如果线程请求的栈深度大于虚拟机栈允许的深度，抛出StackOverflowError异常；如果虚拟机栈可以动态扩展（当前大部分的虚拟机都可以动态扩展），如果扩展时无法申请足够的内存，就会抛出OutOfMemoryError异常

三.本地方法栈

1.与虚拟机栈类似，为Native方法服务（本地方法意味着和平台有关）

四.Java堆

1.被所有线程共享

2.存放对象实例以及数组

3.辣鸡收集器的主要区域

4.堆分为新生代和老年代，再细致一点：Eden空间、Form Survivor空间、To Survivor空间

5.如果堆中没有内存完成实例分配，并且堆也无法扩展，将会抛出outOfMemoryError

五.方法区（栈中的栈帧随着方法的进入和退出进行出栈和入栈操作，每一个栈帧分配多少内存有类结构确定时就确定le）

1.所有线程共享

2.存贮类信息、常量、静态变量、即时编译器编译后的代码

3.运行时常量池（存放编译期生成的各种字面量和符号引用），不一定要求编译期才能产生，运行期间也可能将新的常量放入池中，如String类的intern（）方法

4.通常和永久去perm关联在一起

六.直接内存

1.不是虚拟机运行时数据区的一部分

2.NIO类是一种通道与缓冲区的I/O方式，使用Native函数库直接分配堆外内存，用存贮在java堆中的DirectByteBuffer对象作为引用进行操作

3.显著提高性能，避免了在java堆和Native堆中来回复制数据