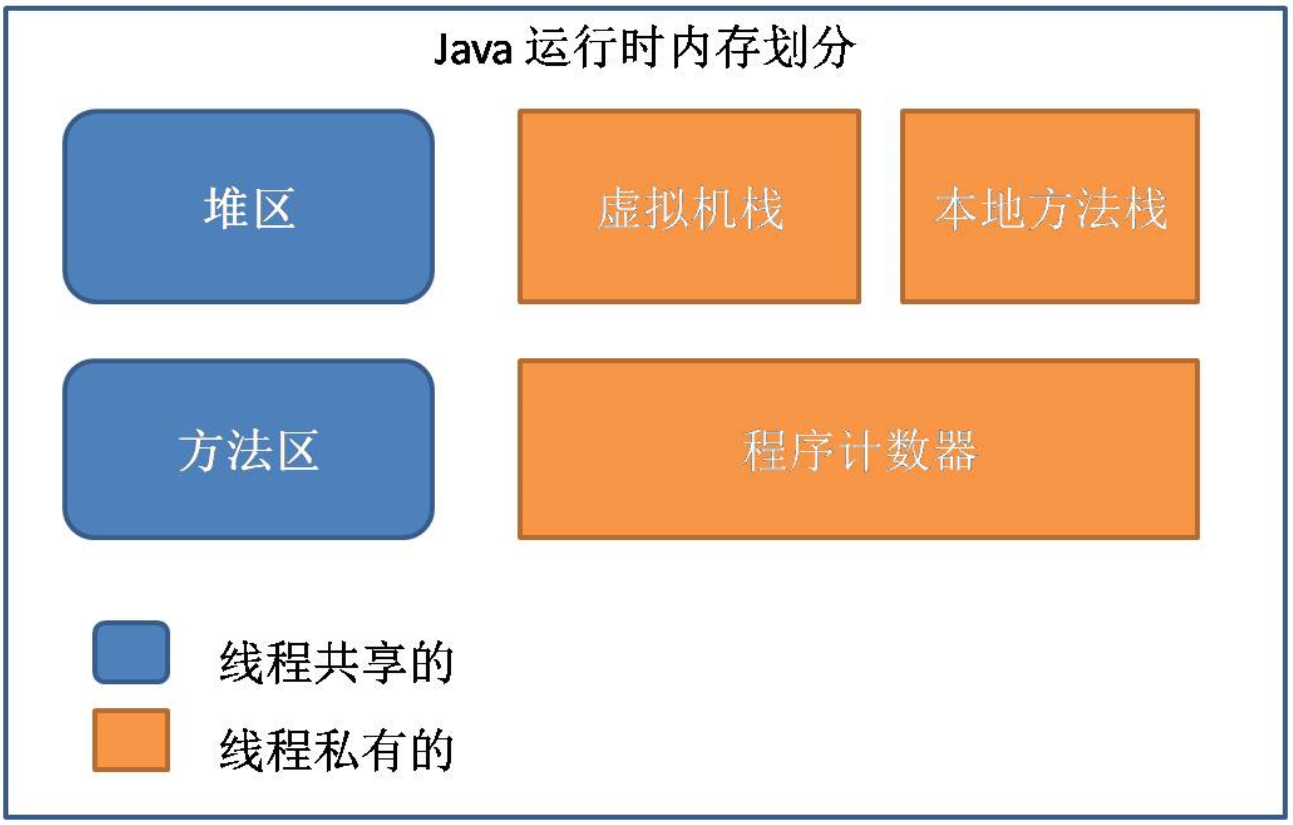
JVM内存

Java内存区域

必须先清楚在JVM中内存区域的划分。在Java运行时的数据区里，由JVM管理的内存区域分为下图几个模块：



其中：

1，程序计数器（Program Counter Register）：程序计数器是一个比较小的内存区域，用于指示当前线程所执行的字节码执行到了第几行，可以理解为是当前线程的行号指示器。字节码解释器在工作时，会通过改变这个计数器的值来取下一条语句指令。

每个程序计数器只用来记录一个线程的行号，所以它是线程私有（一个线程就有一个程序计数器）的。

如果程序执行的是一个Java方法，则计数器记录的是正在执行的虚拟机字节码指令地址；如果正在执行的是一个本地（native，由**C语言**编写 完成）方法，则计数器的值为Undefined，由于程序计数器

只是记录当前指令地址，所以不存在内存溢出的情况，因此，程序计数器也是所有JVM内存区 域中唯一一个没有定义OutOfMemoryError的区域。

2，虚拟机栈（JVM Stack）：一个线程的每个方法在执行的同时，都会创建一个栈帧（Statck Frame），栈帧中存储的有局部变量表、操作站、动态链接、方法出口等，当方法被调用时，栈帧在JVM栈中入栈，当方法执行完成时，栈帧出栈。

局部变量表中存储着方法的相关局部变量，包括各种基本数据类型，对象的引用，返回地址等。在局部变量表中，只有long和double类型会占 用2个局部变量空间（Slot，对于32位机器，一个Slot就是32个bit），其它都是1个Slot。需要注意的是，局部变量表是在编译时就已经确定 好的，方法运行所需要分配的空间在栈帧中是完全确定的，在方法的生命周期内都不会改变。

虚拟机栈中定义了两种异常，如果线程调用的栈深度大于虚拟机允许的最大深度，则抛出StatckOverFlowError（栈溢出）；不过多 数Java虚拟机都允许动态扩展虚拟机栈的大小(有少部分是固定长度的)，所以线程可以一直申请栈，知道内存不足，此时，会抛出 OutOfMemoryError（内存溢出）。

每个线程对应着一个虚拟机栈，因此虚拟机栈也是线程私有的。

3，本地方法栈（Native Method Statck）：本地方法栈在作用，运行机制，异常类型等方面都与虚拟机栈相同，唯一的区别是：虚拟机栈是执行Java方法的，而本地方法栈是用来执行native方法的，在很多虚拟机中（如Sun的JDK默认的HotSpot虚拟机），会将本地方法栈与虚拟机栈放在一起使用。

本地方法栈也是线程私有的。

4，堆区（Heap）：堆区是理解Java GC机制最重要的区域，没有之一。在JVM所管理的内存中，堆区是最大的一块，堆区也是Java GC机制所管理的主要内存区域，堆区由所有线程共享，在虚拟机启动时创建。堆区的存在是为了存储对象实例，原则上讲，所有的对象都在堆区上分配内存（不过现代技术里，也不是这么绝对的，也有栈上直接分配的）。

一般的，根据Java虚拟机规范规定，堆内存需要在逻辑上是连续的（在物理上不需要），在实现时，可以是固定大小的，也可以是可扩展的，目前主 流的虚拟机都是可扩展的。如果在执行垃圾回收之

后，仍没有足够的内存分配，也不能再扩展，将会抛出OutOfMemoryError:Java heap space异常。

关于堆区的内容还有很多，将在下节“Java内存分配机制”中详细介绍。

5，方法区（Method Area）：在Java虚拟机规范中，将方法区作为堆的一个逻辑部分来对待，但事实上，方法区并不是堆（Non-Heap）；另外，不少人的博客中，将Java GC的分代收集机制分为3个代：青年代，老年代，永久代，这些作者将方法区定义为“永久代”，这是因为，对于之前的HotSpot Java虚拟机的实现方式中，将分代收集的思想扩展到了方法区，并将方法区设计成了永久代。不过，除HotSpot之外的多数虚拟机，并不将方法区当做永 久代，HotSpot本身，也计划取消永久代。本文中，由于笔者主要使用Oracle JDK6.0，因此仍将使用永久代一词。

方法区是各个线程共享的区域，用于存储已经被虚拟机加载的类信息（即加载类时需要加载的信息，包括版本、field、方法、接口等信息）、final常量、静态变量、编译器即时编译的代码等。

方法区在物理上也不需要是连续的，可以选择固定大小或可扩展大小，并且方法区比堆还多了一个限制：可以选择是否执行垃圾收集。一般的，方法区上 执行的垃圾收集是很少的，这也是方法区被称为永久代的原因之一（HotSpot），但这也不代表着在方法区上完全没有垃圾收集，其上的垃圾收集主要是针对 常量池的内存回收和对已加载类的卸载。

在方法区上进行垃圾收集，条件苛刻而且相当困难，效果也不令人满意，所以一般不做太多考虑，可以留作以后进一步深入研究时使用。

在方法区上定义了OutOfMemoryError:PermGen space异常，在内存不足时抛出。

运行时常量池（Runtime Constant Pool）是方法区的一部分，用于存储编译期就生成的字面常量、符号引用、翻译出来的直接引用（符号引用就是编码是用字符串表示某个变量、接口的位置，直接引用就是根据符号引用翻译出来的地址，将在类链接阶段完成翻译）；运行时常量池除了存储编译期常量外，也可以存储在运行时间产生的常量（比如String类的intern()方法，作用是String维护了一个常量池，如果调用的字符“abc”已经在常量池中，则返回池中的字符串地址，否则，新建一个常量加入池中，并返回地址）。

6，直接内存（Direct Memory）：直接内存并不是JVM管理的内存，可以这样理解，直接内存，就是JVM以外的机器内存，比如，你有4G的内存，JVM占用了1G，则其余的3G就是直接内存，JDK中有一种基于通道（Channel）和缓冲区 （Buffer）的内存分配方式，将由C语言实现的native函数库分配在直接内存中，用存储在JVM堆中的DirectByteBuffer来引用。 由于直接内存收到本机器内存的限制，所以也可能出现OutOfMemoryError的异常。