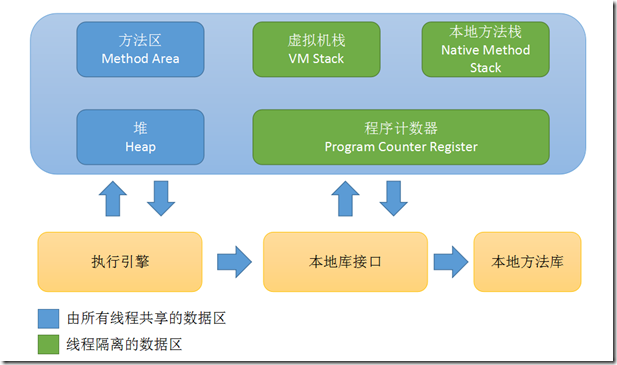
**Java内存区域**

Java虚拟机在执行java程序时，会把它所管理的内存请求划分为几个不同的数据区域。这些区域中有的随着虚拟机进程的启动而创建，有些区域则依赖于用户线程的启动和结束而创建和销毁。Java运行时数据区域如下：



程序计数器(Program Counter Register)[线程私有]

* 程序计数器是一块很小的内存区域，里面存储了当前线程正在执行的字节码指令的地址。
* 因为java虚拟机的多线程是通过线程轮流切换来分配CPU的执行时间，所以，为了线程切换以后能恢复到正确的执行位置，每个线程都有一个独立的程序计数器，各个线程之间的计算器互不影响，独立存储。因此，程序计数器是 “线程私有”的。
* 如果执行的是java方法，这个计数器记录的是正在执行的虚拟机字节码指令地址。如果是native方法，计数器为空。
* 此内存区域是唯一一个在java虚拟机规范中没有规定任何OutOfMemoryError情况的区域。

Java虚拟机栈(Java Virtual Machine stacks) [线程私有]

* Java虚拟机栈用来描述Java方法执行的内存模型,每个方法在执行的同时都会创建一个栈帧(Stack Frame)用于存储局部变量表、操作数栈、动态链接、方法出口等信息。一个方法对应一个栈帧。每一个java方法从调用直至执行结束的过程都对应一个栈帧在虚拟机栈中入栈到出栈的过程。
* 每个线程都有各自独立的java虚拟机栈，虚拟机栈的声明周期和线程一样。
* 规定的异常情况有两种：
  + 1.线程请求的栈的深度大于虚拟机所允许的深度，将抛出StackOverflowError异常；
  + 2.如果虚拟机可以动态扩展，如果扩展时无法申请到足够的内存，就抛出OutOfMemoryError异常。

本地方法栈(Native Method Stack) [线程私有]

* 和java虚拟机栈一样，是“线程私有”的，区别在于，java虚拟机栈是为执行java方法服务的，而本地方法栈是为执行Native方法服务的。
* 本地方法栈也会抛出StackOverFlowException和OutOfMemoryException异常。

Java堆(Java Heap) [所有线程共享]

* 堆是JVM中内存中最大的一块，被所有线程共享的一块内存区域，在JVM创建时创建。
* 堆的作用是存放对象实例。几乎所有的对象实例和对象数组都需要在堆上分配。
* Java堆是垃圾收集器管理的主要区域，也被称之为“GC”堆。
* 如果堆中没有内存完成实例分配，并且堆也无法扩展时，则抛出OutOfMemoryException异常。

方法区(Method Area) [所有线程共享]

* 方法区和java堆一样，是各个线程共享的内存区域，用于存储被虚拟机加载的类信息、常量、静态变量、即时编译器编译的代码等数据。
* 方法区还包含运行时常量池(Runtime Constant Pool)，Class文件中除了有类的版本、字段、方法、接口等描述信息外，还有一项是常量池（Constant Pool Table），用于存放编译期生成的各种字面量和符号引用。并非预置入Class文件中常量池的内容才进入方法运行时常量池，运行期间也可能将新的常量放入池中，这种特性被开发人员利用得比较多的便是String类的intern()方法。
* 当方法区无法满足内存分配需求时，抛出OutOfMemoryError。