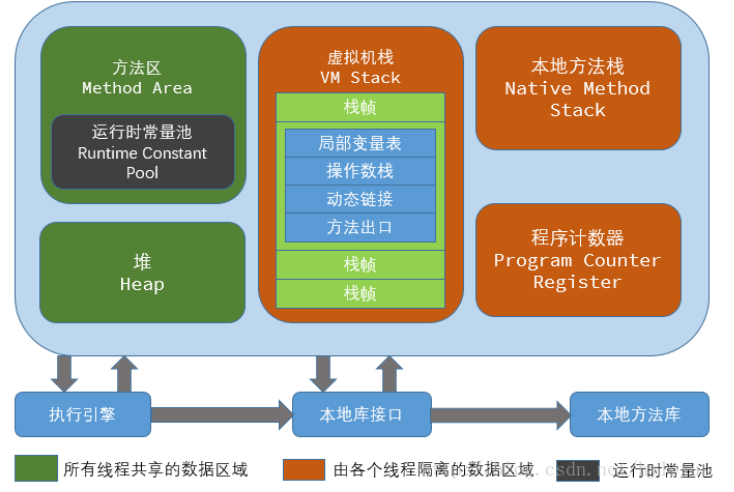
## JVM内存区域

JVM 内存区域主要分为线程私有区域【程序计数器、虚拟机栈、本地方法区】、线程共享区域【JAVA 堆、方法区】、直接内存



### 程序计数器(线程私有)

一块较小的内存空间, 是当前线程所执行的字节码的行号指示器，每条线程都要有一个独立的程序计数器，这类内存也称为“线程私有”的内存。

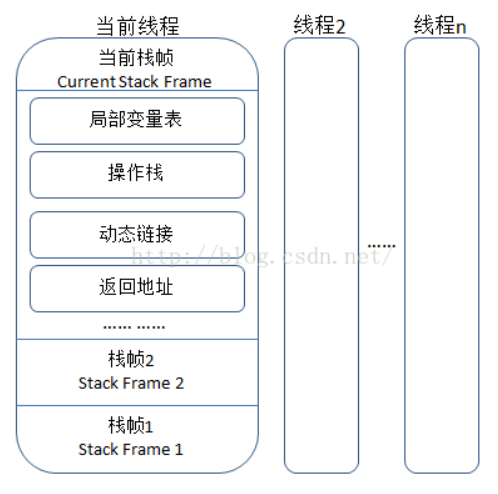
正在执行 java 方法的话，计数器记录的是虚拟机字节码指令的地址（当前指令的地址）。如果还是 Native 方法，则为空。

这个内存区域是唯一一个在虚拟机中没有规定任何 OutOfMemoryError 情况的区域

### 虚拟机栈(线程私有)

是描述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。每一个方法从调用直至执行完成 的过程，就对应着一个栈帧在虚拟机栈中入栈到出栈的过程。

栈帧（ Frame）是用来存储数据和部分过程结果的数据结构，同时也被用来处理动态链接 (Dynamic Linking)、 方法返回值和异常分派（ Dispatch Exception）。栈帧随着方法调用而创建，随着方法结束而销毁——无论方法是正常完成还是异常完成（抛出了在方法内未被捕获的异常）都算作方法结束



#### 局部变量表

局部变量表是一组变量值存储空间，用于存放方法参数和方法内部定义的局部变量；使用的是索引来访问

#### 操作栈

操作数栈也常被称为操作栈，和局部变量区一样，操作数栈也是被组织成一个以字长为单位的数组。通过标准的栈操作—压栈和出栈—来访问的

虚拟机把操作数栈作为它的工作区——大多数指令都要从这里弹出数据，执行运算，然后把结果压回操作数栈

#### 动态链接

当编译java程序的时候，会得到程序汇总每一个类或者接口的独立的class文件，虽然独立看上去毫无关联，但是他们直接通过接口（harbor）符号互相关联，或者与java api的class文件相关联。当运行程序的时候，java虚拟机装载程序的类和接口，并且在动态链接的过程中把他们会想勾连起来，在程序运行中，java虚拟机内存组织了一张互相链接的类和接口的网

#### 返回地址

方法执行前，会保存调用者当前的 PC 计数器中的值，当方法正常退出时，将该 PC 计数器的值会作为返回地址，返回给调用者。在方法异常退出时，返回地址是通过异常处理器表来确定的。

### 本地方法区(线程私有)

本地方法区和 Java Stack 作用类似, 区别是虚拟机栈为执行 Java 方法服务, 而本地方法栈则为Native 方法服务, 如果一个 VM 实现使用 C-linkage 模型来支持 Native 调用, 那么该栈将会是一个C 栈，但 HotSpot VM 直接就把本地方法栈和虚拟机栈合二为一。

### 堆（Heap-线程共享）-运行时数据区

是被线程共享的一块内存区域，创建的对象和数组都保存在 Java 堆内存中，也是垃圾收集器进行垃圾收集的最重要的内存区域。由于现代 VM 采用分代收集算法, 因此 Java 堆从 GC 的角度还可以细分为: 新生代(Eden 区、From Survivor 区和 To Survivor 区)和老年代

### 方法区/永久代（线程共享）

即我们常说的永久代(Permanent Generation), 用于存储被 JVM 加载的类信息、常量、静态变量、即时编译器编译后的代码等数据. HotSpot VM把GC分代收集扩展至方法区, 即使用Java堆的永久代来实现方法区, 这样 HotSpot 的垃圾收集器就可以像管理 Java 堆一样管理这部分内存, 而不必为方法区开发专门的内存管理器(永久带的内存回收的主要目标是针对常量池的回收和类型的卸载, 因此收益一般很小)。

运行时常量池（Runtime Constant Pool）是方法区的一部分。Class 文件中除了有类的版本、字段、方法、接口等描述等信息外，还有一项信息是常量池（Constant Pool Table），用于存放编译期生成的各种字面量和符号引用，这部分内容将在类加载后存放到方法区的运行时常量池中。 Java 虚拟机对 Class 文件的每一部分（自然也包括常量池）的格式都有严格的规定，每一个字节用于存储哪种数据都必须符合规范上的要求，这样才会被虚拟机认可、装载和执行。