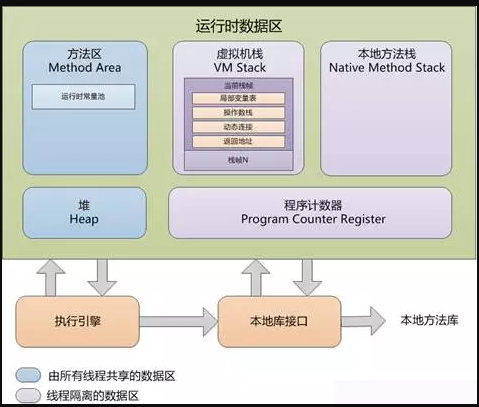
### JVM运行时区域详解



我们知道的JVM内存区域有：堆和栈，这是一种泛的分法，也是按运行时区域的一种分法，堆是所有线程共享的一块区域，而栈是线程隔离的，每个线程互不共享。

# 线程不共享区域

每个线程的数据区域包括程序计数器、虚拟机栈和本地方法栈，它们都是在新线程创建时才创建的。

* **程序计数器（Program Counter Rerister）**

程序计数器区域一块内存较小的区域，它用于存储线程的每个执行指令，每个线程都有自己的程序计数器，此区域不会有内存溢出的情况。

* **虚拟机栈（VM Stack）**

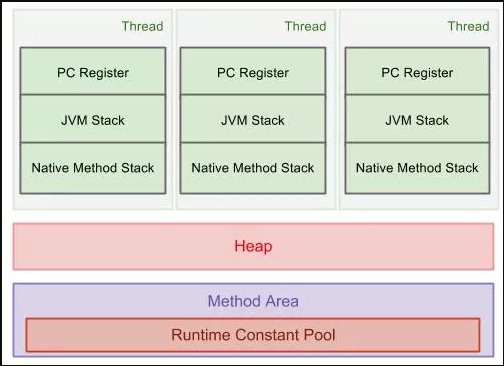
虚拟机栈描述的是Java方法执行的内存模型，每个方法被执行的时候都会同时创建一个栈帧（Stack Frame）用于存储局部变量表、操作数栈、动态链接、方法出口等信息。每一个方法被调用直至执行完成的过程就对应着一个栈帧在虚拟机栈中从入栈到出栈的过程。

* **本地方法栈（Native Method Stack）**

本地方法栈用于支持本地方法（native标识的方法，即非Java语言实现的方法）。

虚拟机栈和本地方法栈，当线程请求分配的栈容量超过JVM允许的最大容量时抛出StackOverflowError异常。

* **线程不共享区域如下图绿色背景所示。**



# **线程共享区域**

线程共享区域包含：堆和方法区。

* **堆（Heap）**

堆是最常处理的区域，它存储在JVM启动时创建的数组和对象，JVM垃圾收集也主要是在堆上面工作。

如果实际所需的堆超过了自动内存管理系统能提供的最大容量时抛出OutOfMemoryError异常。

* **方法区（Method Area）**

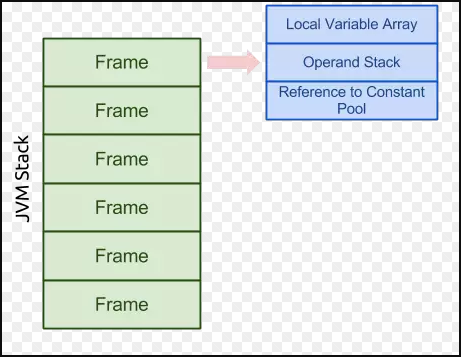
方法区是可供各条线程共享的运行时内存区域。存储了每一个类的结构信息，例如运行时常量池（Runtime Constant Pool）、字段和方法数据、构造函数和普通方法的字节码内容、还包括一些在类、实例、接口初始化时用到的特殊方法。

当创建类和接口时，如果构造运行时常量池所需的内存空间超过了方法区所能提供的最大内存空间后就会抛出OutOfMemoryError

* **运行时常量池（Runtime Constant Pool）**

运行时常量池是方法区的一部分，每一个运行时常量池都分配在JVM的方法区中，在类和接口被加载到JVM后，对应的运行时常量池就被创建。运行时常量池是每一个类或接口的常量池（Constant\_Pool）的运行时表现形式，它包括了若干种常量：编译器可知的数值字面量到必须运行期解析后才能获得的方法或字段的引用。

如果方法区的内存空间不能满足内存分配请求，那Java虚拟机将抛出一个OutOfMemoryError异常。



栈包含Frames，当调用方法时，Frame被推送到堆栈。一个Frame包含局部变量数组、操作数栈、常量池引用。