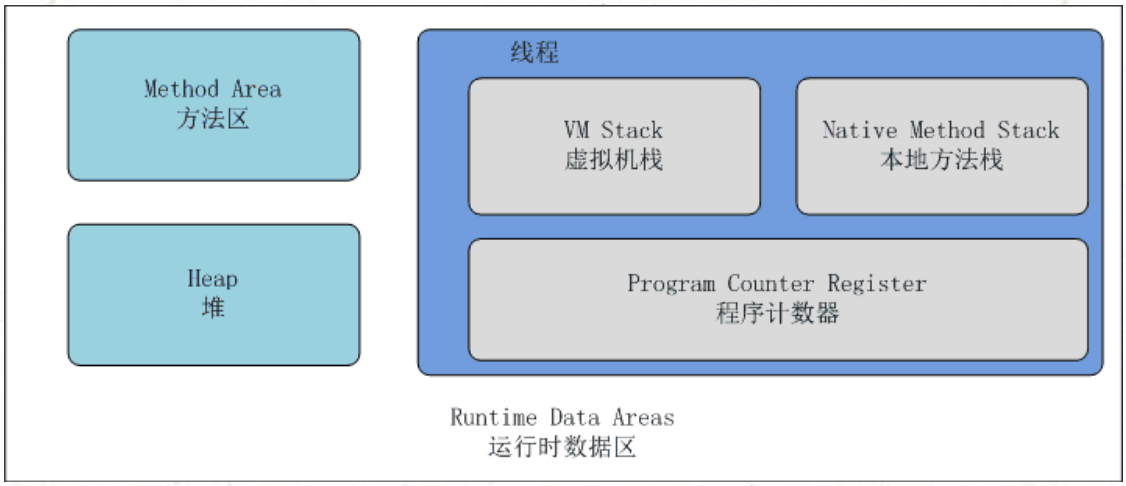
http://www.jianshu.com/p/8dc1c291c972

java语言中不需要像C/C++语言那样需要自己管理内存，内存的申请与释放全部由JVM进行统一管理，这样java中只一个`new`关键字就可以申请内存了。但不是说不用自己管理内存就代表内存不会出现内存泄露了，如果长时间持有对象而不释放很容易造成内存泄露。



java运行时数据区域

通过上图可以知道java运行时数据区域划分为五个部分:

1.方法区

2.堆

3.虚拟机栈

4.本地方法栈

5.程序计数器

方法区 (Method Area)

方法区用于存储已被虚拟机加载的类信息、常量(final修饰, 不可变)、静态变量(static)、即时编译器编译后的代码等数据。

运行时常量池: 方法区的一部分。Class文件中除了有类的版本、字段、方法、接口等描述信息外，还有一项信息是常量池，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进入方法区的运行时常量池中存放。

堆 (Java Heap)

对于大多数应用来说，Java堆是Java虚拟机所管理的内存中最大的一块。Java堆是被所有线程共享的一块内存区域，在虚拟机启动时创建。该区域的唯一目的就是存放对象实例，几乎所有的对象实例都在这里分配内存。该区域也是垃圾收集器的主要区域。

虚拟机栈 (Java Virtual Machine Stacks)

线程私有，生命周期与线程相同。虚拟机栈描述的是Java方法执行的内存模型：每个方法在执行的同时会创建一个**栈帧(Stack Frame)**用于存储局部变量表、操纵数栈、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，就对应一个栈帧在虚拟机中入栈到出栈的过程。

局部变量表：存放了编译器可知的各种基本数据类型，对象引用和returnAddress。

本地方法栈 (Native Method Stack)

与虚拟机栈的作用相似，两者的区别虚拟机栈为虚拟机执行Java方法服务，而本地方法栈为虚拟机用到的Native方法服务。

程序计数器 (Program Counter Register)

一块较小的内存区域，可以看作是当前线程所执行的字节码的行号指示器。在虚拟机的概念模型里，字节码解释器工作时就是通过改变这个计数器的值来选取下一条需要执行的字节码指令，分支、循环、跳转、异常处理、线程回复等基础功能都需要依赖这个计数器来完成。

每条线程都需要有一个独立的程序计数器，各个线程之间计数器互不影响，独立存储。