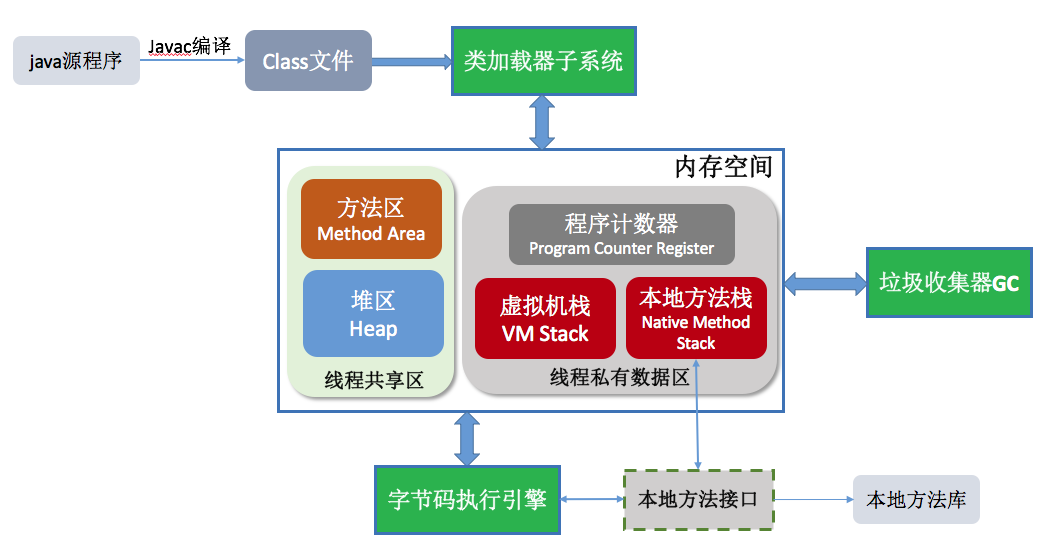
**JVM基本框架**

首先是JVM的整图结构图：   


围绕JVM内存空间有**三个重要的组成部分**（绿色框）：

* [类加载器子系统（Class类文件内部结构怎样？JVM何时加载类？如何加载类？）](http://blog.csdn.net/isunn/article/details/50492582)（另外的博客）
* [字节码执行子引擎 （程序运行时帧栈结构？方法调用时如何支持多态？执行引擎如何工作？)](http://blog.csdn.net/isunn/article/details/50496483) （另外的博客）
* [垃圾回收器GC（如何确定对象可回收？哪些垃圾回收方法？何时进行垃圾回收？）](http://blog.csdn.net/isunn/article/details/50500022)（另外篇博客）

另外的内容还包括：

* Javac编译器（.java源程序如何如何编译成类文件？java的语法糖？）
* 程序的并发执行

**JVM内存空间**

**虚拟机栈**：虚拟机栈描述的是方法Method执行的内存模型：每个方法被调用时都会创建一个栈帧，存放方法局部变量表、操作数栈和动态链接，方法出口地址。栈帧结构如下：   


* + 每个线程都有**私有的栈帧空间，相互独立**。
  + 当前执行方法**拥有顶部当前栈帧，方法的调用和返回过程就是栈帧的入栈和出栈过程。**
  + 局部变量表：**基本数据类型和对象引用局部变量、方法参数和隐含的this指针。**
  + 操作数栈：**JVM字节码指令集是基于操作数栈的**（就像x86硬件指令集基于通用寄存器一样）。
  + 虚拟机栈可通过VM参数-Xss（每个线程栈尺寸）设置栈空间大小，如果当前线程申请的栈深度大于虚拟机允许的深度，StackOverFlowError异常。
* **本地方法栈**：与虚拟机栈类似，只是服务对象是Native方法。
* **程序计数器**：当前线程代码执行指示器，每个线程独立。
* **堆**：几乎所有**对象实例**在堆中分配空间（栈上只是堆中实例的引用），为**所有线程共享**。 （具体可以看另外一篇文章，JVM内存划分）  
  + 堆是垃圾收集器GC的主要管理区域（在讲到GC时会有堆上空间的细分）
  + JIT编辑器的逃逸技术等，使某些对象也能够在栈上分配空间。
* **方法区**：存放**被加载的类信息（方法，属性等）、常量（final类型常量，常量池）、静态变量、即时编译器编译后的本地代码。**
  + HotSpot JVM用**持久代**（Permanet Generation）来存放方法区。
  + 方法区垃圾回收主要是**常量池回收和类型卸载**。

**[运行时常量池]**（Runtime Constant Pool）

* + 是方法区一部分，与Class文件中[常量池]（Constant Pool Table）相对应，当类加载后，Class文件的[常量池]将存放在[运行时常量池]中。
  + [常量池]存放编译期生成的**字面量**（字符串”abc”，声明为final常量等）和**符号引用**（类和接口全限定名、字段名、方法名等）。