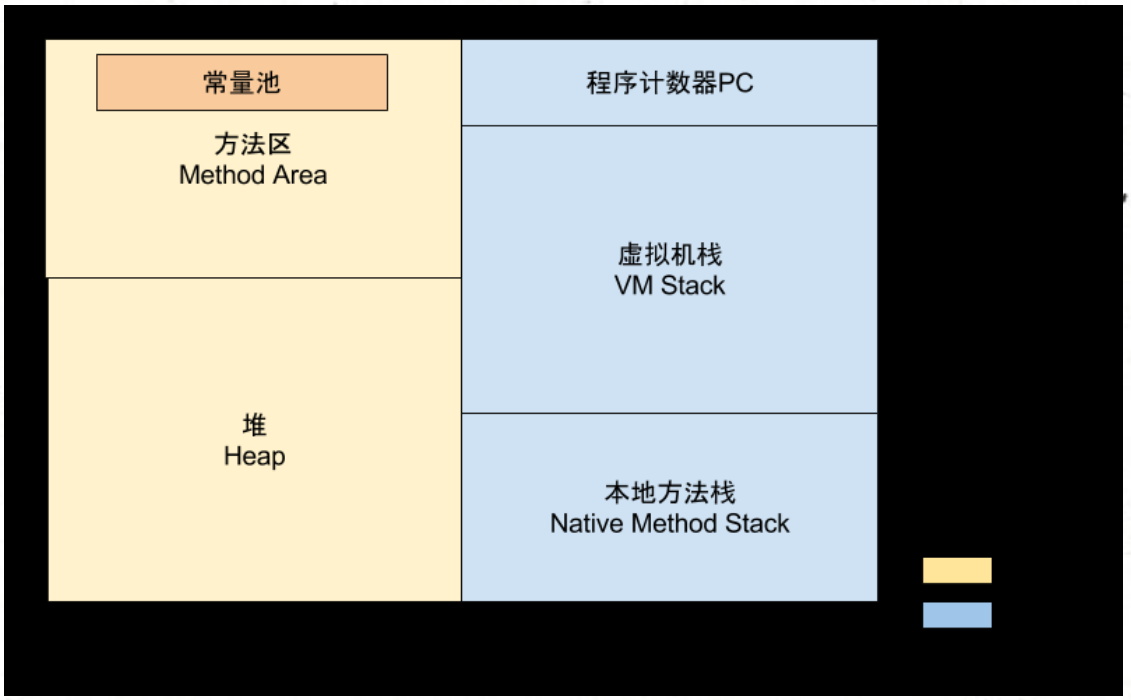
http://gityuan.com/2016/01/09/java-memory/

java是在java虚拟机上运行，一般地大家讲到的Java内存其实就是Jvm内存

一、内存模型

Java内存模型，往往是指Java程序在运行时内存的模型，而Java代码是运行在Java虚拟机之上的，由Java虚拟机通过解释执行(解释器)或编译执行(即时编译器)来完成，故Java内存模型，也就是指Java虚拟机的运行时内存模型。

作为Java开发人员来说，并不需要像C/C++开发人员，需要时刻注意内存的释放，而是全权交给虚拟机去管理，那么有就必要了解虚拟机的运行时内存是如何构成的。运行时内存模型，分为线程私有和共享数据区两大类，其中线程私有的数据区包含程序计数器、虚拟机栈、本地方法区，所有线程共享的数据区包含Java堆、方法区，在方法区内有一个常量池。



（1）线程私有区：

- 程序计数器，记录正在执行的虚拟机字节码的地址；
- 虚拟机栈：方法执行的内存区，每个方法执行时会在虚拟机栈中创建栈帧；
- 本地方法栈：虚拟机的Native方法执行的内存区；

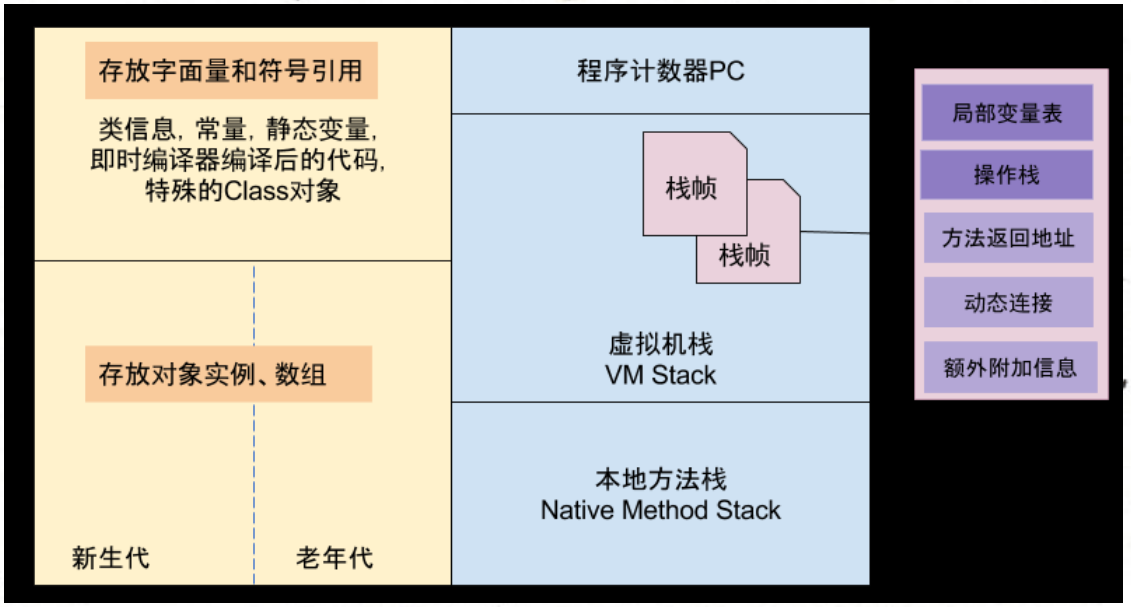
（2）线程共享区：

- Java堆：对象分配内存的区域；
- 方法区：存放类信息、常量、静态变量、编译器编译后的代码等数据；
    - 常量池：存放编译器生成的各种字面量和符号引用，是方法区的一部分。

对于大多数的程序员来说，Java内存比较流行的说法便是堆和栈，这其实是非常粗略的一种划分，这种划分的”堆”对应内存模型的Java堆，”栈”是指虚拟机栈，然而Java内存模型远比这更复杂，想深入了解Java的内存，还是有必要明白整个内存模型。

二、 详细模型

运行时内存分为五大块区域（常量池属于方法区，算作一块区域），前面简要介绍了每个区域的功能，那接下来再详细说明每个区域的内容，Java内存总体结构图如下：



2.1 程序计数器PC

程序计数器PC，当前线程所执行的字节码行号指示器。每个线程都有自己计数器，是私有内存空间，该区域是整个内存中较小的一块。
当线程正在执行一个Java方法时，PC计数器记录的是正在执行的虚拟机字节码的地址；当线程正在执行的一个Native方法时，PC计数器则为空（Undefined）。

2.2 虚拟机栈

虚拟机栈，生命周期与线程相同，是Java方法执行的内存模型。每个方法(不包含native方法)执行的同时都会创建一个栈帧结构，方法执行过程，对应着虚拟机栈的入栈到出栈的过程。

栈帧(Stack Frame)结构

栈帧是用于支持虚拟机进行方法执行的数据结构，是属性运行时数据区的虚拟机站的栈元素。见上图， 栈帧包括：

1. 局部变量表 (locals大小，编译期确定)，一组变量存储空间， 容量以slot为最小单位。
2. 操作栈(stack大小，编译期确定)，操作栈元素的数据类型必须与字节码指令序列严格匹配
3. 动态连接， 指向运行时常量池中该栈帧所属方法的引用，为了 动态连接使用。
    - 前面的解析过程其实是静态解析；
    - 对于运行期转化为直接引用，称为动态解析。
4. 方法返回地址
    - 正常退出，执行引擎遇到方法返回的字节码，将返回值传递给调用者
    - 异常退出，遇到Exception,并且方法未捕捉异常，那么不会有任何返回值。
5. 额外附加信息，虚拟机规范没有明确规定，由具体虚拟机实现。

因此，一个栈帧的大小不会受到

异常(Exception)

Java虚拟机规范规定该区域有两种异常：

- StackOverFlowError：当线程请求栈深度超出虚拟机栈所允许的深度时抛出
- OutOfMemoryError：当Java虚拟机动态扩展到无法申请足够内存时抛出

2.3 本地方法栈

本地方法栈则为虚拟机使用到的Native方法提供内存空间，而前面讲的虚拟机栈式为Java方法提供内存空间。有些虚拟机的实现直接把本地方法栈和

虚拟机栈合二为一，比如非常典型的Sun HotSpot虚拟机。

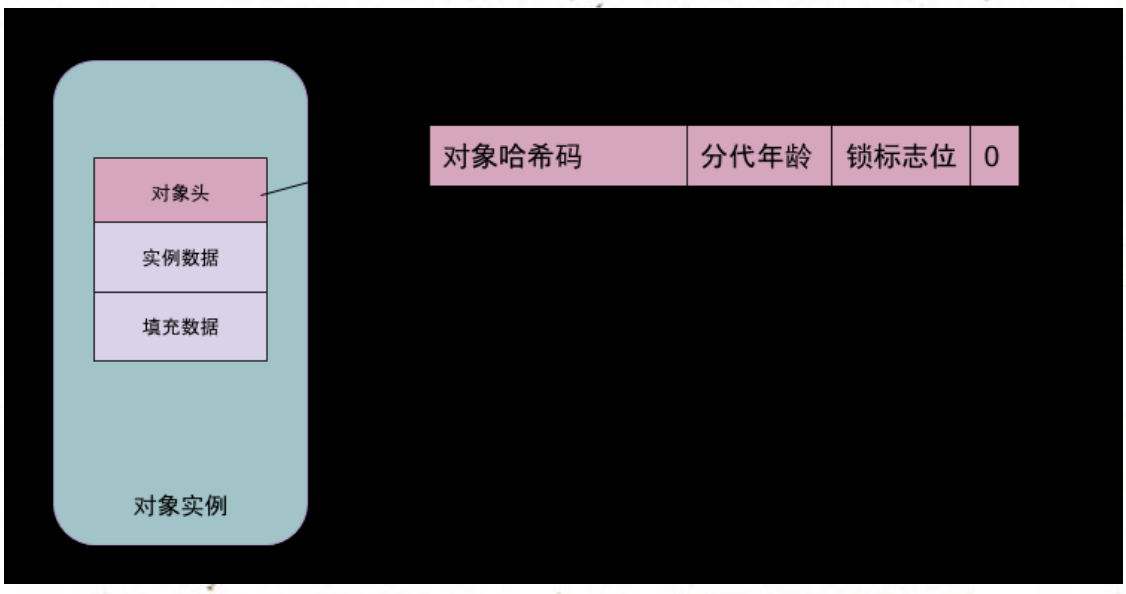
异常(Exception)：Java虚拟机规范规定该区域可抛出StackOverFlowError和OutOfMemoryError。

2.4 Java堆

Java堆，是Java虚拟机管理的最大的一块内存，也是GC的主战场，里面存放的是几乎所有的对象实例和数组数据。JIT编译器有栈上分配、标量替换等优化技术的实现导致部分对象实例数据不存在Java堆，而是栈内存。

- 从内存回收角度，Java堆被分为新生代和老年代；这样划分的好处是为了更快的回收内存；
- 从内存分配角度，Java堆可以划分出线程私有的分配缓冲区(Thread Local Allocation Buffer,TLAB)；这样划分的好处是为了更快的分配内存；

对象创建的过程是在堆上分配着实例对象，那么对象实例的具体结构如下：



对于填充数据不是一定存在的，仅仅是为了字节对齐。HotSpot VM的自动内存管理要求对象起始地址必须是8字节的整数倍。对象头本身是8的倍数，当对象的实例数据不是8的倍数，便需要填充数据来保证8字节的对齐。该功能类似于高速缓存行的对齐。

另外，关于在堆上内存分配是并发进行的，虚拟机采用CAS加失败重试保证原子操作，或者是采用每个线程预先分配TLAB内存.

异常(Exception)：Java虚拟机规范规定该区域可抛出OutOfMemoryError。

2.5 方法区

方法区主要存放的是已被虚拟机加载的类信息、常量、静态变量、编译器编译后的代码等数据。GC在该区域出现的比较少。

异常(Exception)：Java虚拟机规范规定该区域可抛出OutOfMemoryError。

2.6 运行时常量池

运行时常量池也是方法区的一部分，用于存放编译器生成的各种字面量和符号引用。运行时常量池除了编译期产生的Class文件的常量池，还可以在运行期间，将新的常量加入常量池，比较常见的是String类的intern()方法。

- 字面量：与Java语言层面的常量概念相近，包含文本字符串、声明为final的常量值等。
- 符号引用：编译语言层面的概念，包括以下3类：
  - 类和接口的全限定名
  - 字段的名称和描述符
  - 方法的名称和描述符

但是该区域不会抛出OutOfMemoryError异常。