# 一．Java 内存区域 运行时数据区

## 1.程序计数器

每个线程都有独立的程序计数器，各线程之间计数器互不影响，独立储存，属于“线程私有”的内存区域

遇到跳转指令会改变这个寄存器的值，从而把程序流“导向”到一个指定的存储空间地址上，if/while/switch等等跳转语句以及函数调用、中断历程就是这么回事。

执行java方法，计数器记录的是执行的虚拟机字节码指令的地址；执行native方法，计数器值为空

Java虚拟机规范中程序计数器是唯一没有规定任何OutOfMemory情况的区域。**原因**是因为程序计数器存储的数据所占的空间大小不会随程序执行而发生改变。

## 2.Java 虚拟机栈

Java 虚拟机栈也是线程私有的，生命周期与线程相同。方法执行的同时会创建栈帧（虚拟机栈的栈元素），栈帧主要用于存放局部变量表和操作数栈。

在局部变量表（区）中，方法的局部变量以及方法的参数就存在这张表中。存放了编译器可知的各种基本数据类型、对象引用（reference类型，它不等同于对象本身，可能是一个指向对象起始地址的引用指针，也可能是指向一个代表对象的句柄或其他与此对象相关的位置）和returnAddress类型（指向了一条字节码指令的地址）。

Long、double占两个局部变量空间（Slot），其余数据类型只占用一个

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。但是和前者不同的是，它不是通过索引来访问，而是通过标准的栈操作—压栈和出栈—来访问的。比如，如果某个指令把一个值压入到操作数栈中，稍后另一个指令就可以弹出这个值来使用。

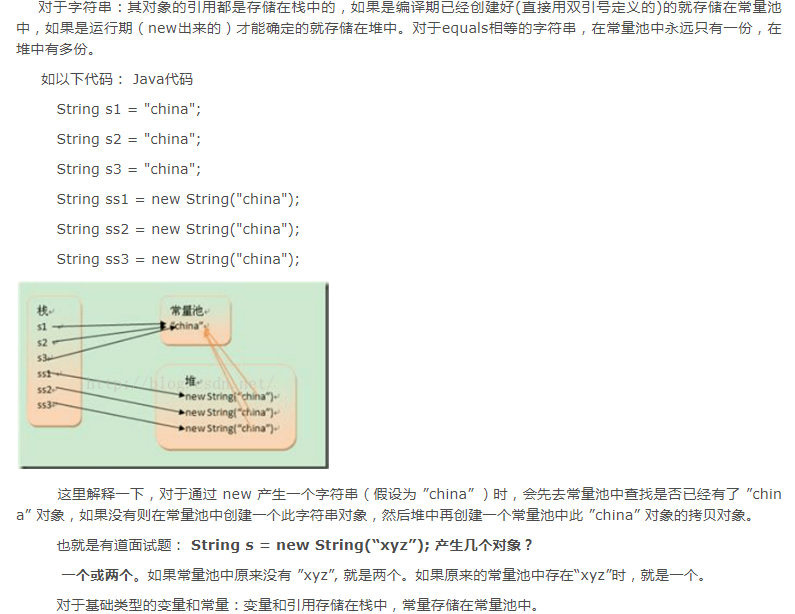
这个区域中，线程请求的栈深度大于虚拟机允许的深度，就抛出StackOverflowError异常；如果虚拟机栈扩展时无法申请到足够内存时，就会OutOfMemoryError。

## 3.Java 本地方法栈

与虚拟机栈发挥的作用非常相似，区别是虚拟机栈执行Java方法（字节码服务）服务，本地方法栈则为虚拟机使用到的Native方法服务。

## 4.Java 堆

Java堆（Java Heap）是被所有线程共享的一块区域，在虚拟机启动时创建。用来存放对象实例。GC管理器主要作用于这块区域



堆中没有内存完成实例分配，且堆无法再扩展时，将会抛出OutOfMemoryError异常。

## 5.方法区

方法区与Java堆一样，是各个线程共享的内存区域，用于存储已被虚拟机加载的类信息、常量、静态变量、即时编译器编译后的代码等数据，别名（Non-Heap）非堆。方法区无法满足内存分配需求时，抛出OutOfMemoryError异常。

## 6.运行时常量池

是方法区的一部分，用于存放编译期生成的各种字面量和符号引用。

# 二．对象

## 1.对象的创建

## 2.对象的内存分布

## 3.对象的访问定位

# 三．垃圾收集器与内存分配策略

## 1.概述

由于程序计数器、虚拟机栈、本地方法栈3个区域随线程生灭（内存分配基本在编译期就确定），这几个区域不需要多考虑垃圾回收，栈帧随着方法的进入和退出做入栈和出栈操作，实现了自动的内存清理，因此，我们的内存垃圾回收主要集中于 java 堆和方法区中，在程序运行期间，这部分内存的分配和使用都是动态的。