JAVA内存区域与内存溢出异常

1.1、程序计数器（线程私有）

首先必须明确一点，程序计数器是一个线程私有的一块内存空间，这个内存空间相对较小，表示的是当前进程所执行的字节码的行号指示器，字节码解释器工作时就是通过改变这个计数器的值选取下一条需要执行的字节码指令，每一个线程都有一个独立的程序计数器。这类的内存也被称之为“内存私有”的内存。

程序计数器存在的必要：由于java虚拟机的多线程是通过线程轮流切换、分配处理器执行时间的方式来实现的，为了线程切换后能恢复到正确的执行位置，每条线程都需要有一个独立的程序计数器。因为各个线程之间的计数器互不影响，独立存储。所以我们称这类内存区域为“线程私有”的内存。

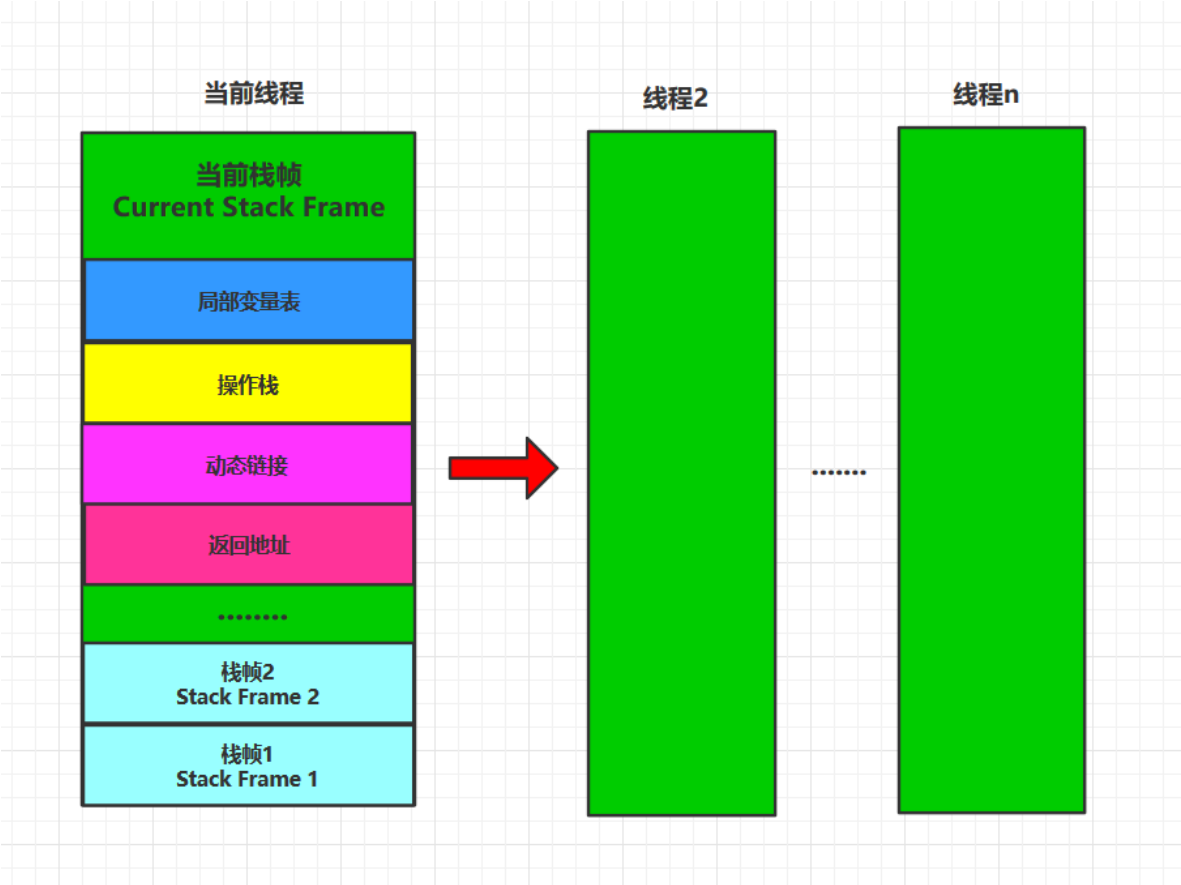
如果执行的是本地方法（Native）方法，这个计数器的值则应为空（Undefined）。

解释：本地方法是C/C++暴露给java的一个接口。

注：此内存区域是*唯一一个*在《java虚拟机规范》中没有规定任何OutOfMemoryError情况的区域。

1.2、虚拟机栈（线程私有）

是描述Java方法执行的内存模型，每个方法在执行的同时都会创建一个栈帧（Stack Frame）用于存储局部变量、操作数栈、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，就对应着一个栈帧在虚拟机栈中入栈到出栈的过程。



栈帧（Stack Frame）是用来存储数据和部分过程结果的数据结构，同时也被用来处理动态链接（Dynamic Linking）、方法返回值和异常分派（Dispatch Exception）。栈帧随着方法调用而创建，随着方法结束而销毁---无论方法是完成还是异常完成（抛出了在方法内未被捕获的异常）都算作方法结束。

局部变量表存放了编译期间可知的各种Java虚拟机基本数据类型（boolean、byte、char、short、short、float、long、double）、对象引用（reference类型。注意：此处的对象引用仅仅是指向对象相关位置的引用指针）。基本数据类型在java虚拟机中的存储空间以局部变量槽（slot）来表示，其中64位长度的long与double类型数据是使用两个slot来表示，而其他的数据类型都是使用一个slot来表示。

注：当栈的深度超过虚拟机所允许的深度时，会抛出StackOverflowError异常。当栈无法申请到足够的存储会抛出OutOfMemoryError异常。

1.3本地方法栈（线程私有）

本地方法栈与虚拟机栈的作用很类似，区别的是本地方法栈是为Native方法服务。在HotSpot VM中，直接将本地方法栈与虚拟机栈合二为一。

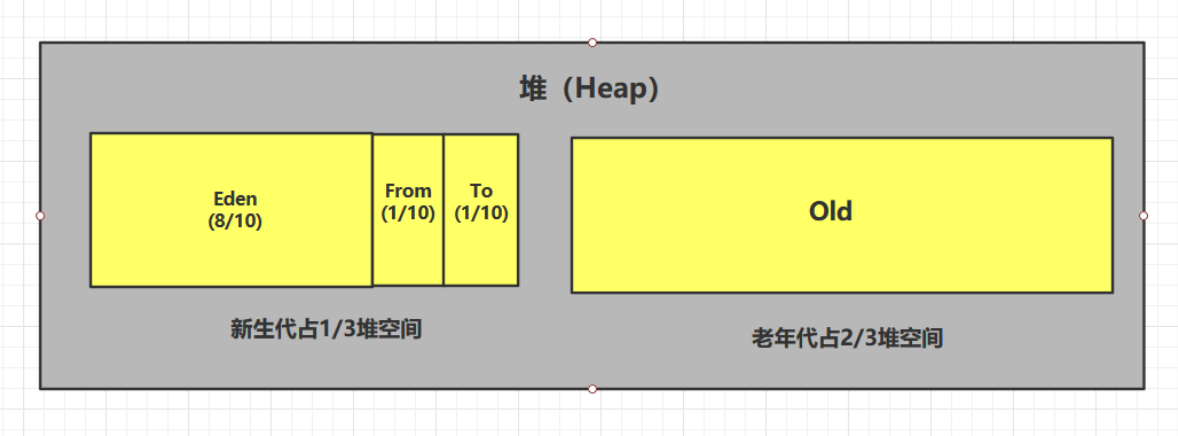
与虚拟机栈一样，当栈的深度超过虚拟机所允许的深度时，会抛出StackOverflowError异常。当栈无法申请到足够的存储会抛出OutOfMemoryError异常。

1.4 Java堆（线程共享）

java堆是被线程共享的一块内存区域，创建对的对象以及数组都保存再Java堆内存中，也是垃圾收集器进行垃圾收集的最重要的内存。Java世界中几乎所有的对象实例都是在这里分配内存。

从内存分配的角度看，所有线程共享的Java堆中可以划分多个线程私有的分配缓冲区（TLAB），以提升对象分配时的效率。

Java堆从GC的角度还可以细分为新生代和老年代



1.5.1新生代中有Eden区，From区，To区

新生代：是用来存放新生的对象。一般占堆的3/1空间。由于频繁的创建对象，因此新生代会频繁地触发MinorGC进行垃圾回收。

Eden区:Java新对象的出生地（如果新创建的对象占用内存很大，则直接分配到老年代）。当Eden区的内存不够时会触发MinorGC对Eden区进行一次垃圾回收。

ServivorFrom：上次GC的幸存者，作为这一次GC的被扫描者

ServivorTo保留了一次MinorGC过程中的幸存者。

MinorGC的执行过程：复制-->清空-->互换

MinorGC采用的是复制算法：

1.首先，把Eden和ServivorFrom区域中存活的对象复制到ServivorTo区域（如果有对象的年龄以及达到了老年的标准，则赋值到老年代），同时把这些对象的年龄+1（如果ServivorTo不够位置了，也会被放到老年代）

2.清空Eden、ServivorFrom中的对象

3.ServivorTo和ServivorFrom中的对象互换，原ServivorTo称为下一次GC的ServivorFrom区。

1.5.2老年代

注：当堆无法再扩展的时候，会抛出OutOfMemoryError异常

1.5 方法区（线程共享）

方法区用于存储被JVM加载的类信息、常量、静态变量、及时编译器编译后的代码等数据。HotSpot虚拟机把GC分代收集扩展至方法区，即使用Java堆的永久代来实现方法，因此许多Java程序员习惯把永久代与方法区划等号，在《深入理解Java虚拟机》一书中，作者有提到这两者并非划等号。永久代只是HotSpot虚拟机中使用的一个概念。

HotSpot中使用永久代来实现方法区，使得HotSpot的垃圾收集器就可以像管理Java堆一样管理这部分内存，而不必为方法区开发专业的内存管理器（永久代的内存回收主要目标是针对常量池的回收和类型的卸载，因此收益一般很小）

1.6运行时常量池

运行时常量池（Runtime Constant Pool）是方法区的一部分。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Runtime Constant Pool。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存放到方法区的运行时常量池中。Java虚拟机对Class文件的每一部分的格式都有严格的规定，每一个字节用于存储哪种数据都必须符合规范上的要求，这样才会被虚拟机任课、装载和执行。

注：当常量池无法再申请到内存时会抛出OutOfMemoryError异常