# JVM内存模型

东方奇骥

1. 内存模型



1. 程序计数器

程序计数器是一块较小的内存空间，可以看作当前线程所执行的字节码的行号指示器。字节码解释器工作时，通过改变程序计数器的值**选取下一条需要执行的字节码指令**，例如，分支、循环、跳转、异常、线程恢复等都依赖于计数器。

程序计数器是唯一不会出现 OutOfMemoryError 的内存区域。

1. 方法区

方法区主要是用来存放已被虚拟机加载的**类相关信息**，包括类信息、运行时常量池、字符串常量池、静态变量。类信息又包括了类的版本、字段、方法、接口和父类等信息。

方法区与堆空间类似，也是一个共享内存区，所以方法区是线程共享的。假如两个线程都试图访问方法区中的同一个类信息，而这个类还没有装入 JVM，那么此时就只允许一个线程去加载它，另一个线程必须等待。

**运行时常量池**

运行时常量池是方法区的一部分。Class 文件中除了有类的版本、字段、方法、接口等描述信息外，还有常量池信息，用于存放编译器生成的字面量和符号引用，这些信息将在类加载后存放到方法区的运行时常量池中。

运行时常量池也受到方法区内存的限制，当常量池无法再申请到内存时将抛出 OutOfMemoryError 异常。

**直接内存（堆外内存）**

直接内存不是虚拟机运行时数据区域的一部分，也不是虚拟机规范中定义的内存区域，但是这部分也被频繁地使用，而且也可能导致 OutOfMemoryError 异常出现。

本机直接内存的分配不受到 Java 堆大小的限制，但是直接内存仍然受到本机总内存地大小及处理器寻址空间的限制。如果各个内存区域的总和大于物理内存限制，就会导致动态扩展时出现 OutOfMemoryError 异常。

1. 虚拟机栈

Java 虚拟机栈是线程私有的内存空间，它和 Java 线程一起创建。当创建一个线程时，会在虚拟机栈中申请一个线程栈，用来保存方法的局部变量、操作数栈、动态链接方法和返回地址等信息，并参与方法的调用和返回。每一个方法的调用都伴随着栈帧的入栈操作，方法的返回则是栈帧的出栈操作。

Java 虚拟机栈会出现两种异常。

如果虚拟机栈不可以动态扩展，当线程请求的栈深度大于虚拟机所允许的深度时，将抛出 StackOverflowError 异常；如果虚拟机栈可以动态扩展，当无法申请到足够的内存时，将抛出 OutOfMemoryError 异常。

1. 本地方法栈

本地方法栈跟 Java 虚拟机栈的功能类似，Java 虚拟机栈用于管理 Java 函数的调用，而本地方法栈则用于管理本地方法的调用。但本地方法并不是用 Java 实现的，而是由 C 语言实现的。

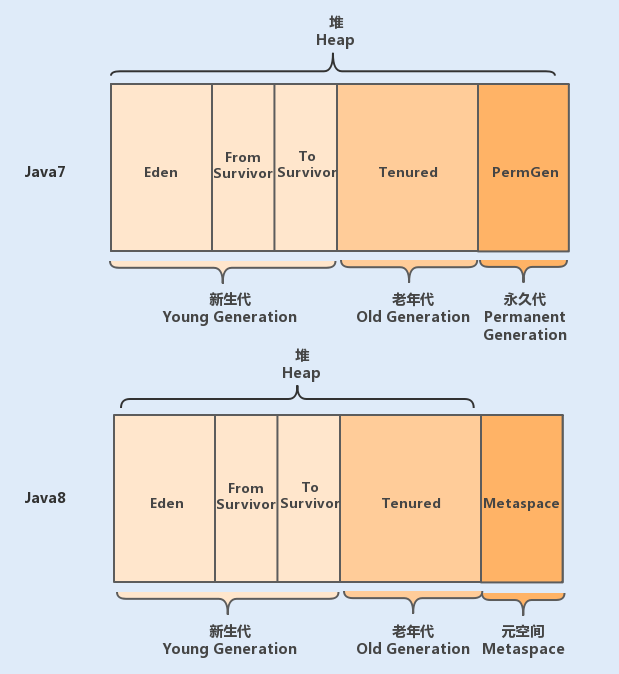
和虚拟机栈一样，本地方法栈也会出现 StackOverflowError 和 OutOfMemoryError 两种异常。

1. 堆

堆是 Java 虚拟机管理的内存中最大的一块，其目的是存放对象实例，几乎所有的对象实例都在堆中分配内存。

见后一节解读。

1. Java7堆VS Java8堆



在 HotSpot 虚拟机、Java7 版本中已经将永久代的静态变量和运行时常量池转移到了堆中，其余部分则存储在 JVM 的非堆内存中，而 Java8 版本已经将方法区中实现的永久代去掉了，并用元空间（class metadata）代替了之前的永久代，并且元空间的存储位置是本地内存（堆外内存）。之前永久代的类的元数据存储在了元空间，永久代的静态变量（class static variables）以及运行时常量池（runtime constant pool）则跟 Java7 一样，转移到了堆中。

为什么要用元空间代替永久代？

1）移除永久代是为了融合 HotSpot JVM 与 JRockit VM 而做出的努力，因为 JRockit 没有永久代，所以不需要配置永久代。

2）永久代内存经常不够用或发生内存溢出，爆出异常 java.lang.OutOfMemoryError: PermGen。这是因为在 JDK1.7 版本中，指定的 PermGen 区大小为 8M，由于 PermGen 中类的元数据信息在每次 FullGC 的时候都可能被收集，回收率都偏低，成绩很难令人满意；还有，为 PermGen 分配多大的空间很难确定，PermSize 的大小依赖于很多因素，比如，JVM 加载的 class 总数、常量池的大小和方法的大小等。

一定要注意，上面的模型分区，只是逻辑上的概念。JDK8, 认为静态变量和运行时常量以及类信息在方法区中，是说的逻辑概念。实际物理内存看，静态变量和运行时常量在堆内存中，而类信息在元空间（堆外内存）。