链滴

Java - JVM

作者：someone33881

原文链接：https://ld246.com/article/1545481834802

来源网站：链滴

    本文主要是记录在学习
ava - JVM过程中的一些知识点备忘！

知识点规整：

一、

- 介绍下Java内存区域（运行时数据区）：程序计数器、虚拟机栈、本地方法栈、堆、方法区、运行常量池、直接内存
- Java对象的创建过程：五步，需要能够默认出来并且知道每一步虚拟机都做了些什么
- 对象的访问定位的两种方式：句柄、直接指针
- String类和常量池
- 8种基本类型的包装类和常量池

20181223-1

1、运行时数据区域

Java虚拟机在执行java程序的过程中会把它管理的内存划分成若干个不同的数据区域：有些区域是线私有的（程序计数器、虚拟机栈、本地方法栈），有些区域是线程共享的（堆、方法区、直接内存）

1.1、程序计数器

程序计数器是一块较小的内存空间，可以看作是当前线程锁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这个计数器的值来选取下一条需要执行的字节码指令，分支、循环、跳、异常处理、线程恢复等功能都需要依赖这个计数器来完成！

另外，为了线程切换后能够恢复到正确的执行位置，每个线程都需要一个独立的程序计数器，各个线之间计数器互不影响，独立存储，因此这类内存区域也被称为"线程私有"的内存！

程序计数器的作用主要有两个：

● 字节码解释器通过改变程序计数器来依次读取指令，从而实现 **代码的流程控制**，如：顺序执行、择、循环、异常处理

● 在多线程的情况下，程序计数器用于记录当前线程执行的位置，从而当 **线程切换**回来的时候能够道该线程上次运行到哪儿了

注意：程序计数器是唯一一个不会出现OutOfMemoryError的内存区域，它的生命周期随着线程的创而创建，随着线程的结束而死亡

1.2、java虚拟机栈

与程序计数器一样，java虚拟机栈也是线程私有的，它的生命周期与线程相同，描述的是java方法执的内存模型

java内存可以粗略低区分为堆内存（Heap）和栈内存（Stack），其中栈也即现在说的虚拟机栈，或说是虚拟机栈中局部变量表部分（实际上，java虚拟机栈是由一个个栈帧组成，而每个栈帧都拥有：部变量表、操作数栈、动态链接、方法出口信息）

局部变量表主要用于存放编译器可知的各种数据类型（boolean、byte、char、short、int、float、long、double）、对象引用（reference类型，它不同于对象本身，可能是一个指向对象起始地址的引指针，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）

java虚拟机栈会出现两种异常：StackOverFlowError和OutOfMemoryError

● StackOverFlowError：若java虚拟机栈的内存大小 **不允许动态扩展**，那么当线程**请求栈的深度**超当前java虚拟机栈的最大深度的时候，则抛出该异常

● OutOfMemoryError：若java虚拟机栈的内存大小 **允许动态扩展**，那么当线程**请求栈时内存用完了**无法再动态扩展了，则会抛出该异常

java虚拟机栈也是线程私有的，每个线程均有各自的java虚拟机栈，而且随着线程的创建而创建，随线程的死亡而死亡

1.3 本地方法栈

与虚拟机栈所发挥的作用非常相似，区别在于：虚拟机栈为虚拟机执行java方法（也就是字节码）服，而本地方法栈则为虚拟机使用到的Native方法服务！在HotSpot虚拟机中，本地方法栈与Java虚拟栈合二为一！

本地方法被执行的时候，在本地方法栈也会创建一个栈帧，用于存放该本地方法的局部变量表、操作栈、动态链接、出口信息

方法执行完毕后相应的栈帧也会出栈并释放内存空间，也会出现StackOverFlowError和OutOfMemoyError两种异常

1.4、堆

Java虚拟机所管理的内存中最大的一块，java堆是所有线程共享的一块内存区域，在虚拟机启动时创！此内存区域的唯一目的就是存放对象实例，几乎所有的对象实例以及数组均在这里分配内存

Java堆是垃圾回收器管理的主要区域，因此也被称作GC堆！从垃圾回收角度出发，由于现在垃圾收器基本上都是采用分代垃圾收集算法，因此java堆也还可以细分为“新生代和老年代（以及jdk1.7之还有：永久代）”，再细致一点新生代可以细分“Eden空间、From Survivor、To Survivor空间等，进一步划分的目的是为了更好地回收内存或者说更快地分配内存

在JDK1.8中移除了整个永久代，取而代之的是一个叫元空间（Metaspace）的区域（永久代使用的是VM的堆内存空间，而元空间使用的是物理内存，直接受到本机的物理内存限制）

1.5、方法区

方法区与java堆一样，是各个线程共享的内存区域，它用于存储已被虚拟机加载的类信息、常量、静变量、即时编译器编译后的代码等数据！

虽然java虚拟机规范把方法区描述为堆的一个逻辑部分，但是它却有一个别名Non-Heap(非堆)，目是为了与java堆区分开来

HotSpot虚拟机中方法区也常被称为“永久代”，本质上两者并不等价！仅仅是因为HotSpot虚拟机计团队用永久代来实现方法区而已，这样HotSpot虚拟机的垃圾收集器就可以像管理java堆一样管理部分内存了！但这并不是一个好主意，因为这样更容易遇到内存溢出问题

相对而言，垃圾收集行为在这个区域是比较少出现的，但并非数据进入方法区后就“永久存在”了

1.6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，有常量池信息（用于存放编译期生成各种字面量和符合引用）

既然运行时常量池是方法区的一部分，自然受到方法区内存的限制，当常量池无法再申请到内存时会出OutOfMemoryError异常

JDK1.7+版本JVM已经将运行时常量池从方法区中移了出来，在Java堆中开辟了一块区域存放运行时量池

1.7、直接内存

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

JDK1.4中新加入的NIO（New Input/Output）类，引入了一种基于通道Channel与缓存区Buffer的I/方式，它可以直接使用Native函数库直接分配堆外内存，然后通过一个存储在Java堆中的DirectByteBffer对象作为这块内存的引用进行操作！！这样能够在一些场景中显著提高性能，因为避免了在Java和Native堆之间来回复制数据

本机直接内存的分配不会收到Java堆的限制，但是，既然是内存就会受到本机总内存大小以及处理器址空间的限制