# Java知识图谱

# Java虚拟机

Java（java virtual machine）虚拟机是执行java字节码的平台，通俗的讲就是一个程序。

## 运行时数据区

Java虚拟机在执行java程序的时候会把他所管理的内存划分为若干个不同的数据区域。这些区域有不同的用处以及有不同的创建时机。运行时数据区包括，程序计数器、java虚拟机栈、本地方法栈、堆、方法区（内含运行时常量池）、直接内存（非虚拟机管理的区域，但是会被经常使用）

程序计数器

方法区

堆

Java虚拟机栈

本地方法栈

执行引擎

本地方法

### 程序计数器

程序计数器是java运行时的一块较小的内存区域，它属于线程私有，随着线程的创建而创建，随着线程的销毁而销毁。它代表当前字节码执行的行号指示器，字节码解释器的工作就是通过这个行号指示器来选举下一条需要执行的指令、分支、跳转、异常以及线程切换等。如果执行的是java方法，此时代表的是字节码的行号，如果执行的是Native方法则程序计数器的值为空。此区域也是java虚拟机唯一标识没有OutOfMemoryError的区域。

### 虚拟机栈

与程序计数器一样，虚拟机栈也是线程私有的区域，它和线程的生命周期相同。虚拟机栈描述的是java方法执行的内存模型，每个方法执行的时候都会创建一个栈帧用来存储局部变量表，操作数栈，方法的出入空信息。每个方法的调用就对应着栈帧的入栈和出栈的过程。

#### 局部变量表

局部变量表存放了编译时可知的数据类型,包含基本类型（byte、char、int、long、double、float、boolean）和对象引用（reference 可能是对象的引用指针或者是句柄）returnAdress（指向一条字节码指令的地址）。其中64位长度的long/double 占用两个局部变量表的空间其余的占用一个空间。局部变量表所需的空间在编译的时候就已经确定了。当进入一个方法的时候在栈帧中分配多大的局部变量表示确定的，不会在运行时期改变。

在java虚拟机规范中对此区域定义了两种异常情况，当线程请求的栈的深度大于虚拟机栈所允许的栈的深度的时候会出现StackOverflowError。当线程请求所需内存大于虚拟机栈所分配的内存的时候会出现OutOfMemoryError异常。

### 本地方法栈

本地方法栈和虚拟机栈一样也是线程私有的区域，只不过本地方法栈运行的是Native方法，与虚拟机栈一样本地方法栈也会抛出StackOverflowError和OutOfMemoryError。

### 堆

堆是java虚拟机管理的最大的一块区域，它是被线程共享的一块区域，在虚拟机创建的时候创建，此区域唯一的目的是存放对象的实例，数组对象等。java堆是垃圾收集器管理的主要区域，基于垃圾收集算法的不同，堆划分为新生代和老年代。java虚拟机规范了此区域可以处于物理上不连续的空间，只要是逻辑上连续即可。在此区域中可能发生OutOfMemoryError异常，堆的大小是可控的使用-XX:max20m -XX:mas20m 来调整堆的大小。

### 方法区

方法区和堆一样也是属于线程共享的一块区域，他用于存储已被虚拟机加载的类信息、常量、静态变量以及及时编译后的代码数据等信息。虽然java虚拟机把方法区描述为堆的一部分，但是它确是“非堆”对于再SunHotspot虚拟机上开发的人员来说此区域被称作永久代。

### 运行时常量池

运行时常量池属于方法区的一部分，用于存放编译期生产的字面量和符号引用。这部分内容将在运行时进入运行时常量池中。运行时常量池相对于Class常量池具备了动态性，也就是并非预置在Class文件中的常量池才能进入方法区的运行时常量池，运行期间也能将新的常量放入运行时常量池中，典型的是String.intern()方法。运行时常量池收方法区的约束，当无法再申请到足够的内存后会抛出OutOfMemoryError异常。

### 直接内存

直接内存并不是java虚拟机所管理的一块内存区域，有与jdk1.4之后引入了NIO（基于通道的IO）它可以使Native函数直接分配堆外内存。此区域收物理内存的影响，当内存不足的时候同样会出现OutOfMemoryError异常。

## 什么是对象

对象是存储再java堆中的Class实例。

### 对象的创建

java是一门面向对象的语言，在程序运行的过称重无时无刻不有对象创建出来，可以通过 new 关键字，反序列化，反射等。那在虚拟机底层是如何创建对象的呢？

虚拟机遇到一个new 指令时会去检查这个指令的参数是否在常量池中定位到一个符号引用并且检查这个符号引用的类是否已经加载解析初始化过，如果没有那就先进行类的加载。