链滴

简述 jvm 原理与工作流程

作者：lbaron

原文链接：https://ld246.com/article/1594886996094

来源网站：链滴

<h3 id="1--内存模型">1. 内存模型</h3>
1) 方法区 - 持久代（Permanet Generation （线程共享）
a. 保存方法代码(编译后的 java 代码)和符号表。
b. 存放了要加载的类信息、静态变量、final 类型的常量、属性和方法信息。
c. 可通过-XX:PermSize 和-XX:MaxPermSize 来指定最小值和最大值。
2) 堆 （线程共享）
a.存放实例化后对象的内存。
b.通过-Xmx 和-Xms 来控制。
c.有操作系统空闲链表遍历寻找第一个空间大于申请空间的堆节点，然后从空闲链表中移除。空间大申请空间的部分，系统自动放回空闲链表。
3) 虚拟机栈
a.存放基本类型的变量,对象的引用和方法调用
b.先进后出与后进先出的规则
4) java 本地栈
虚拟机栈为虚拟机执行 Java 方法（字节码）服务，而本地方法栈则为虚拟机使用到的 Native 方法服。
5) 程序计数器
每个线程都有一个程序计算器，就是一个指针，指向方法区中的方法字节码（下一个将要执行的指令码），由执行引擎读取下一条指令
<h3 id="2--类加载机制">2. 类加载机制</h3>
1) Bootstrap ClassLoader
负责加载 $JAVA_HOME 中 jre/lib/rt.jar 里所有的 class，由 C++ 实现，不是 ClassLoader 子类

2) Extension ClassLoader
负责加载 java 平台中扩展功能的一些 jar 包，包括 $JAVA_HOME 中 jre/lib/*.jar 或-Djava.ext.dirs 定目录下的 jar 包
3) App ClassLoader
负责记载 classpath 中指定的 jar 包及目录中 class
4) Custom ClassLoader
属于应用程序根据自身需要自定义的 ClassLoader，如 tomcat、jboss 都会根据 j2ee 规范自行实现 ClassLoader
<h3 id="3--类的执行机制">3. 类的执行机制</h3>
JVM 执行 class 字节码，线程创建后，都会产生程序计数器（PC）和栈（Stack），程序计数器放下一条要执行的指令在方法内的偏移量，栈中存放一个个栈帧，每个栈帧对应着每个方法的每次调，而栈帧又是有局部变量区和操作数栈两部分组成，局部变量区用于存放方法中的局部变量和参数，作数栈中用于存放方法执行过程中产生的中间结果。
<h3 id="4--类的实例化过程">4. 类的实例化过程</h3>

子类的静态字段

<ol start="2">
子类的静态构造方法

<ol start="3">
子类的实例字段

<ol start="4">
父类的静态字段

<ol start="5">
父类的静态构造方法

<ol start="6">
父类的实例字段

<ol start="7">
父类的实例构造方法

<ol start="8">
子类的实例构造方法

<h3 id="5--gc回收机制">5. gc 回收机制</h3>
GC 将内存分为 年轻代、Survivor、eden
1) 标记-删除算法
首先标记出所有需要回收的对象，标记完成后统一回收所有被标记的对象。
2) 标记-复制算法
将可用的内存分为两块，每次只用其中一块，当这一块内存用完了，就将还存活着的对象复制到另外块上面，然后再把已经使用过的内存空间一次性清理掉。
3) 标记整理算法
让所有存活对象都向一端移动，然后直接清理掉边界以外的内存。
4) 分代收集算法
基于前面三种算法的结合
<h3 id="6--内存泄漏与内存溢出">6. 内存泄漏与内存溢出</h3>
内存溢出：（out of memory）通俗理解就是内存不够，通常在运行大型件或游戏时，软件或游戏所需要的内存远远超出了你主机内安装的内存所承受大小，就叫内存溢出。br>
内存泄漏：（Memory Leak）是指程序中己动态分配的堆内存由于某种原因程未释放或无法释放，造成系统内存的浪费，导致程序运行速度减慢甚至系统崩溃等严重后果。
常见的内存泄漏有：
1)静态集合类像 HashMap、Vector 等
2)各种连接，数据库连接，网络连接，IO 连接等没有显示调用 close 关闭，不被 GC 回收导致内存泄。
3)监听器的使用，在释放对象的同时没有相应删除监听器的时候也可能导致内存泄露。