链滴

jvm 类加载过程

作者：dreamertn9527
原文链接：https://ld246.com/article/1515566706850
来源网站：链滴

<h3 id="jvm类加载过程">jvm 类加载过程</h3>
<blockquote>
jvm 类初始化过程分为五个阶段，分别是加载、连接、初始化、使用、卸载，其中连接阶段又分三个阶段，分别是验证、准备、解析三个阶段
</blockquote>
<h4 id="加载">加载</h4>
在加载阶段，虚拟机主要完成以下几个事情：

通过一个类的全限定名称类获取此类的二进制字节流，字节流的获取方式不仅仅是通过 class 文，可以是网络，动态生成，数据库等
将这个字节流所代表的静态存储结构转化为方法区的运行时数据结构
在内存中生成一个 Class 对象，作为方法区的这个类的入口

加载阶段和连接阶段（Linking）的部分内容（如一部分字节码文件格式验证动作）是交叉进行，加载阶段尚未完成，连接阶段可能已经开始，但这些夹在加载阶段之中进行的动作，仍然属于连接段的内容，这两个阶段的开始时间仍然保持着固定的先后顺序。
<h4 id="连接">连接</h4>
连接分为三个阶段，分别是验证、准备、解析
<h5 id="验证">验证</h5>
验证是连接阶段的第一步，目的是确保加载的 class 文件字节流的安全，以及符合当前虚拟机的求
a. 文件格式验证：验证版本号、是否以 0XCAFFBABY 开头、常量池中的常量是否有不支持的类
b. 元数据验证：对字节码进行语义分析，以保障其描述的信息符合 Java 语言规范的要求
c. 字节码验证：通过控制流、数据流进行分析，保障程序的语义是合法的、符合逻辑的
d. 符号引用验证：确保解析动作能够正确执行
<h5 id="准备">准备</h5>
准备阶段是正式为类变量分配内存并设置类变量初始值的阶段，这些变量所使用的内存都将在方区中进行分配。这时候进行内存分配的仅包括类变量（被 static 修饰的变量），而不包括实例变量，例变量将会在对象实例化时随着对象一起分配在堆中。其次，这里所说的初始值“通常情况”下是数类型的零值，假设一个类变量的定义为：
<pre><code class="language-java highlight-chroma">public static int value=123;
</code></pre>
那变量 value 在准备阶段过后的初始值为 0 而不是 123.因为这时候尚未开始执行任何 java 方法而把 value 赋值为 123 的 putstatic 指令是程序被编译后，存放于类构造器()方法之中，所以把 value 赋值为 123 的动作将在初始化阶段才会执行。
至于“特殊情况”是指：public static final int value=123，即当类字段的字段属性是 ConstantValu 时，会在准备阶段初始化为指定的值，所以标注为 final 之后，value 的值在准备阶段初始化为 123 非 0.
<h5 id="解析">解析</h5>
解析阶段是虚拟机将常量池内的符号引用替换为直接引用的过程。解析动作主要针对类或接口、段、类方法、接口方法、方法类型、方法句柄和调用点限定符 7 类符号引用进行。
<h4 id="初始化">初始化</h4>
类初始化阶段是类加载过程的最后一步，到了初始化阶段，才真正开始执行类中定义的 java 程代码。在准备极端，变量已经付过一次系统要求的初始值，而在初始化阶段，则根据程序猿通过程序定的主管计划去初始化类变量和其他资源，或者说：初始化阶段是执行类构造器()方法的过程.
()方法是由编译器自动收集类中的所有类变量的赋值动作和静态语句块 static{}中的语句合并产生的，译器收集的顺序是由语句在源文件中出现的顺序所决定的，静态语句块只能访问到定义在静态语句块前的变量，定义在它之后的变量，在前面的静态语句块可以赋值，但是不能访问。
虚拟机规范严格规定了有且只有 5 中情况（jdk1.7）必须对类进行“初始化”（而加载、验证、

备自然需要在此之前开始）：</p>

遇到 new,getstatic,putstatic,invokestatic 这类调字节码指令时，如果类没有进行过初始化，则要先触发其初始化。生成这 4 条指令的最常见的 Java 代码场景是：使用 new 关键字实例化对象的时、读取或设置一个类的静态字段（被 final 修饰、已在编译器把结果放入常量池的静态字段除外）的候，以及调用一个类的静态方法的时候。
使用 java.lang.reflect 包的方法对类进行反射调用的时候，如果类没有进行过初始化，则需要先发其初始化。
当初始化一个类的时候，如果发现其父类还没有进行过初始化，则需要先触发其父类的初始化。
当虚拟机启动时，用户需要指定一个要执行的主类（包含 main()方法的那个类），虚拟机会先初化这个主类。
当使用 jdk1.7 动态语言支持时，如果一个 java.lang.invoke.MethodHandle 实例最后的解析结 REF_getstatic,REF_putstatic,REF_invokeStatic 的方法句柄，并且这个方法句柄所对应的类没有进初始化，则需要先出触发其初始化。
