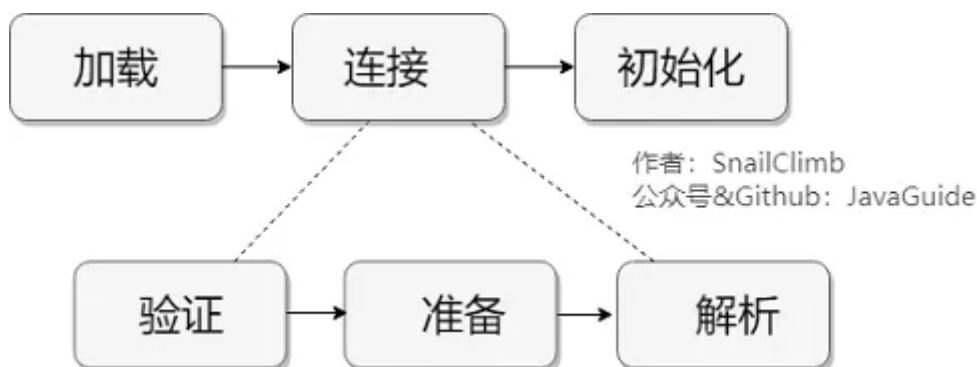


class文件需要加载到虚拟机中之后才能运行和使用，那么虚拟机是如何加载这些class文件的呢？

系统记载class类型的文件主要三步：**加载 -> 连接 -> 初始化**

连接过程又可以分为三步：**验证 -> 准备 -> 解析**



加载

类加载过程的第一步，主要完成下面3件事情

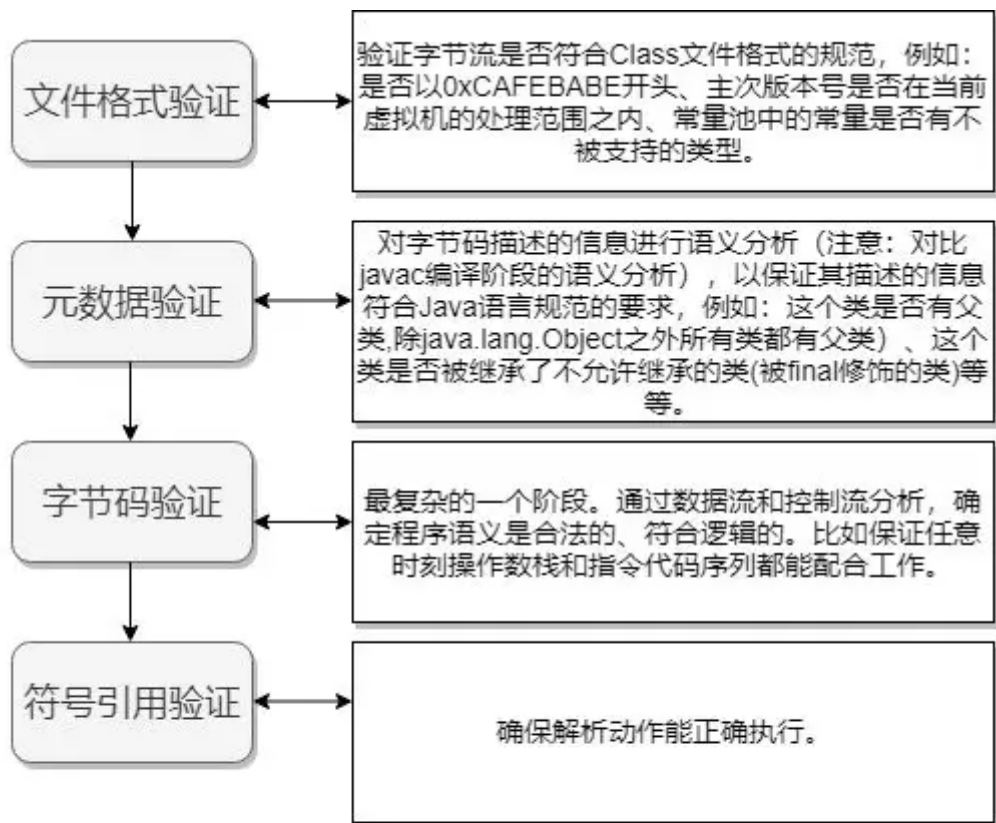
1. 通过全类名获取定义此类的二进制字节流
2. 将字节流所代表的静态存储结构转换为方法区的运行时数据结构
3. 在内存中生成一个代表该类的class对象，作为方法区这些数据的访问入口

虚拟机规范多上面这3点并不具体，因此是非常灵活的。比如："通过全类名获取定义此类的二进制字节流" 并没有指明具体从哪里获取、怎样获取。比如：比较常见的就是从 ZIP 包中读取（日后出现的JAR、EAR、WAR格式的基础）、其他文件生成（典型应用就是JSP）等等。

一个非数组类的加载阶段（加载阶段获取类的二进制字节流的动作）是可控性最强的阶段，这一步我们可以去完成还可以自定义类加载器去控制字节流的获取方式（重写一个类加载器的 `loadClass()` 方法）。数组类型不通过类加载器创建，它由 Java 虚拟机直接创建。

加载阶段和连接阶段的部分内容是交叉进行的，加载阶段尚未结束，连接阶段可能就已经开始了。

验证



作者：SnailClimb
公众号&Github：JavaGuide

准备

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，这些内存都将在方法区中分配。对于该阶段有以下几点需要注意：

1. 这时候进行内存分配的仅包括类变量（static），而不包括实例变量，实例变量会在对象实例化时随着对象一块分配在Java堆中。
2. 这里所设置的初始值"通常情况"下是数据类型默认的零值（如0、0L、null、false等），比如我们定义了public static int value=111，那么value变量在准备阶段的初始值就是0而不是111（初始化阶段才会复制）。特殊情况：比如给value变量加上了final关键字public static final int value=111，那么准备阶段value的值就被复制为111。

基本数据类型的零值：

数据类型	零 值	数据类型	零 值
int	0	boolean	false
long	0L	float	0.0f
short	(short) 0	double	0.0d
char	'\u0000'	reference	null
byte	(byte) 0		

基本数据类型的零值

解析

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、字段、类方法、接口方法、方法类型、方法句柄和调用限定符7类符号引用进行。

符号引用就是一组符号来描述目标，可以是任何字面量。**直接引用**就是直接指向目标的指针、相对偏移量或一个间接定位到目标的句柄。在程序实际运行时，只有符号引用是不够的，举个例子：在程序执行方法时，系统需要明确知道这个方法所在的位置。Java 虚拟机为每个类都准备了一张方法表来存放类中所有的方法。当需要调用一个类的方法的时候，只要知道这个方法在方法表中的偏移量就可以直接调用该方法了。通过解析操作符号引用就可以直接转变为目标方法在类中方法表的位置，从而使得方法可以被调用。

综上，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，也就是得到类或者字段、方法在内存中的指针或者偏移量。

初始化

初始化是类加载的最后一步，也是真正执行类中定义的 Java 程序代码(字节码)，初始化阶段是执行类构造器 `<clinit> ()` 方法的过程。

对于 `<clinit> ()` 方法的调用，虚拟机会自己确保其在多线程环境中的安全性。因为 `<clinit> ()` 方法是带锁线程安全，所以在多线程环境下进行类初始化的话可能会引起死锁，并且这种死锁很难被发现。

对于初始化阶段，虚拟机严格规范了有且只有5中情况下，必须对类进行初始化：

1. 当遇到 `new`、`getstatic`、`putstatic`或`invokestatic` 这4条直接码指令时，比如 `new` 一个类，读取一个静态字段(未被 `final` 修饰)、或调用一个类的静态方法时。
2. 使用 `java.lang.reflect` 包的方法对类进行反射调用时，如果类没初始化，需要触发其初始化。
3. 初始化一个类，如果其父类还未初始化，则先触发该父类的初始化。
4. 当虚拟机启动时，用户需要定义一个要执行的主类(包含 `main` 方法的那个类)，虚拟机会先初始化这个类。
5. 当使用 JDK1.7 的动态动态语言时，如果一个 `MethodHandle` 实例的最后解析结构为 `REF_getStatic`、`REF_putStatic`、`REF_invokeStatic`、的方法句柄，并且这个句柄没有初始化，则需要先触发器初始化。