虚拟机类加载机制

虚拟机把描述类的数据从Class文件加载到内存，并对数据进行校验、转换解析和初始化，最终形成可以被Java虚拟机直接使用的Java类型，这就是虚拟机的类加载机制。

类从被加载到虚拟内存中开始，到卸载内存为止，它的整个生命周期包括了：加载(Loading)、验证(Verification)、准备(Preparation)、解析(Resolution)、初始化(Initialization)、使用(Using)和卸载(Unloading)七个阶段。其中，验证，准备和解析三个部分统称为连接(Linking)。

类加载的过程

类加载的全过程，加载，验证，准备，解析和初始化这五个阶段。

加载

在加载阶段，虚拟机需要完成以下三件事情：

- 通过一个类的全限定名来获取定义此类的二进制字节流
- 将这个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
- 在Java堆中生成一个代表这个类的java.lang.Class对象，作为方法区这些数据的访问入口

验证

这一阶段的目的是为了确保Class文件的字节流中包含的信息符合当前虚拟机的要求，并且不会危害虚拟机自身的安全。不同的虚拟机对类验证的实现可能有所不同，但大致上都会完成下面四个阶段的检验过程：文件格式验证、元数据验证、字节码验证和符号引用验证。

文件格式验证

第一阶段要验证字节流是否符合Class文件格式的规范，并且能被当前版本的虚拟机处理。

元数据验证

第二阶段是对字节码描述的信息进行语义分析，以保证其描述的信息符合Java语言规范的要求。

字节码验证

第三阶段时整个验证过程中最复杂的一个阶段，主要工作是数据流和控制流的分析。在第二阶段对元数据信息中的数据类型做完校验后，这阶段将对类的方法体进行校验分析。这阶段的任务是保证被校验类的方法在运行时不会做出危害虚拟机安全的行为。

符号引用验证

最后一个阶段的校验发生在虚拟机将符号引用直接转化为直接引用的时候，这个转化动作将在连接的第三个阶段－解析阶段产生。符号引用验证可以看作是对类自身以外（常量池中的各种符号引用）的信息进行匹配性的校验。

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进行分配。

解析

解析阶段是虚拟机将常量池的符号引用转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、接口方法四类符号引用进行。

- 类或接口的解析
- 字段解析
- 类方法解析
- 接口方法解析

初始化

前面的类加载过程中，除了在加载阶段用户应用程序可以通过自定义类加载器参与之外，其余动作完全由Java虚拟机主导和控制。到了初始化阶段，才真正开始执行类中定义的Java程序代码（或者说是字节码）。在准备阶段，变量已经赋过一次系统要求的初始值，而在初始化阶段，则是根据程序员通过程序制定的主观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。

类加载器

类与类加载器

虚拟机设计团队把类加载阶段中的"通过一个类的全限定名来获取描述此类的二进制字节流"这个动作放到Java虚拟机外部去实现，以便让程序自己决定如何去获取所需的类。实现这个动作的代码模块被称为"类加载器"。

双亲委派模型

站在Java虚拟机的角度讲，只存在两种不同的类加载器：一种是启动类加载器(Bootstrap ClassLoader)，这个类加载器使用C++语言实现，是虚拟机自身的一部分；另外一种就是所有其他的类加载器，这些类加载器都由Java语言实现，独立于虚拟机外部，并且全部继承自抽象类java.lang.ClassLoader。从Java开发人员的角度来看，类加载器还可以分得更细致一些，绝大部分Java程序都会使用到以下三种系统提供的类加载器：

- 启动类加载器
- 扩展类加载器
- 应用程序类加载器