在JVM的类加载过程中，在解析阶段，Java虚拟机会把类的二级制数据中的符号引用替换为直接引用。

**符号引用**

符号引用以一组符号来描述所引用的目标，符号可以是任何形式的字面量，只要使用时能够无歧义的定位到目标即可。例如，在Class文件中它以CONSTANT\_Class\_info、CONSTANT\_Fieldref\_info、CONSTANT\_Methodref\_info等类型的常量出现。符号引用与虚拟机的内存布局无关，引用的目标并不一定加载到内存中。在Java中，一个java类将会编译成一个class文件。在编译时，java类并不知道所引用的类的实际地址，因此只能使用符号引用来代替。比如org.simple.People类引用了org.simple.Language类，在编译时People类并不知道Language类的实际内存地址，因此只能使用符号org.simple.Language（实际中是由类似于CONSTANT\_Class\_info的常量来表示的）来表示Language类的地址。各种虚拟机实现的内存布局可能有所不同，但是它们能接受的符号引用都是一致的，因为符号引用的字面量形式明确定义在Java虚拟机规范的Class文件格式中。

**直接引用**

直接引用是和虚拟机的布局相关的，同一个符号引用在不同的虚拟机实例上翻译出来的直接引用一般不会相同。如果有了直接引用，那引用的目标必定已经被加载入内存中了。

直接引用可以有不同的实现方式：

* 直接指向目标的指针（比如，指向“类型”【Class对象】、类变量、类方法的直接引用可能是指向方法区的指针）
* 相对偏移量（比如，指向实例变量、实例方法的直接引用都是偏移量）
* 一个能间接定位到目标的句柄

作者：平头哥2015  
链接：https://www.jianshu.com/p/33d7495f9253  
来源：简书  
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。