[**java -- JVM的符号引用和直接引用**](http://www.cnblogs.com/shinubi/articles/6116993.html)

**在JVM中类加载过程中，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。**

1.符号引用（Symbolic References）：

　　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](http://lib.csdn.net/base/javaee)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（假设是这个，当然实际中是由类似于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。

2.直接引用：

 直接引用可以是

（1）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）

（2）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）

（3）一个能间接定位到目标的句柄

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。

[RednaxelaFX](https://www.zhihu.com/people/rednaxelafx)的解释：

作者：RednaxelaFX

链接：https://www.zhihu.com/question/50258991/answer/120450561

来源：知乎

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。

我了解了调用函数时符号引用如何转换为直接引用的,但是对于类变量，实例变量的解析方法还是不太清楚。

符号引用是只包含语义信息，不涉及具体实现的；而解析（resolve）过后的直接引用则是与具体实现息息相关的。所以当谈及某个符号引用被resolve成怎样的直接引用时，必须要结合某个具体实现来讨论才行。

   Java类从加载到虚拟机内存中开始，到卸载出内存为止，它的整个生命周期包括，加载 ,验证 , 准备 , 解析 , 初始化 , 卸载 ，总共七个阶段。其中验证 ,准备 , 解析 统称为连接。

   而在解析阶段会有一个步将常量池当中二进制数据当中的符号引用转化为直接引用的过程。

符号引用 ：符号引用以一组符号来描述所引用的目标。符号引用可以是任何形式的字面量，只要使用时能无歧义地定位到目标即可，符号引用和虚拟机的布局无关。个人理解为：在编译的时候一个每个java类都会被编译成一个class文件，但在编译的时候虚拟机并不知道所引用类的地址，多以就用符号引用来代替，而在这个解析阶段就是为了把这个符号引用转化成为真正的地址的阶段。

public class Test{

   public static void main() {

     String s=”adc”;

     System.out.println(“s=”+s);

   }

}

这个代码中在编译的时候对应的s会被解析成为符号引用,

public class Test{

   public static void main() {

     System.out.println(“s=”+”abc”);

   }

}

而这段代码执行的时候会直接解析成直接引用。

直接引用 ：直接引用和虚拟机的布局是相关的，不同的虚拟机对于相同的符号引用所翻译出来的直接引用一般是不同的。如果有了直接引用，那么直接引用的目标一定被加载到了内存中。

直接引用可以是：

1：直接指向目标的指针。（个人理解为：指向对象，类变量和类方法的指针）

2：相对偏移量。      （指向实例的变量，方法的指针）

3：一个间接定位到对象的句柄。

————————————————

版权声明：本文为CSDN博主「小-鸟」的原创文章，遵循CC 4.0 by-sa版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/qq\_34402394/article/details/72793119