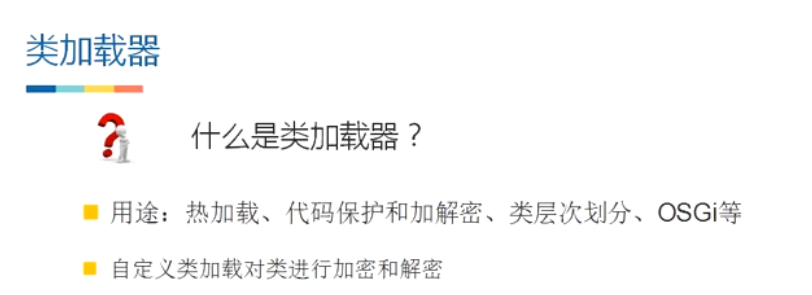
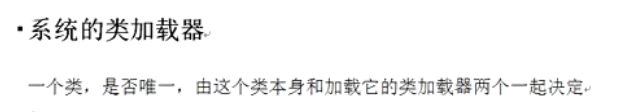
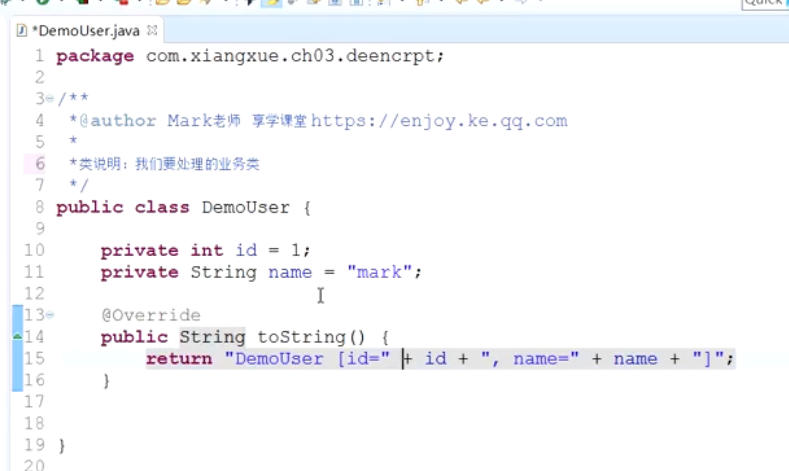


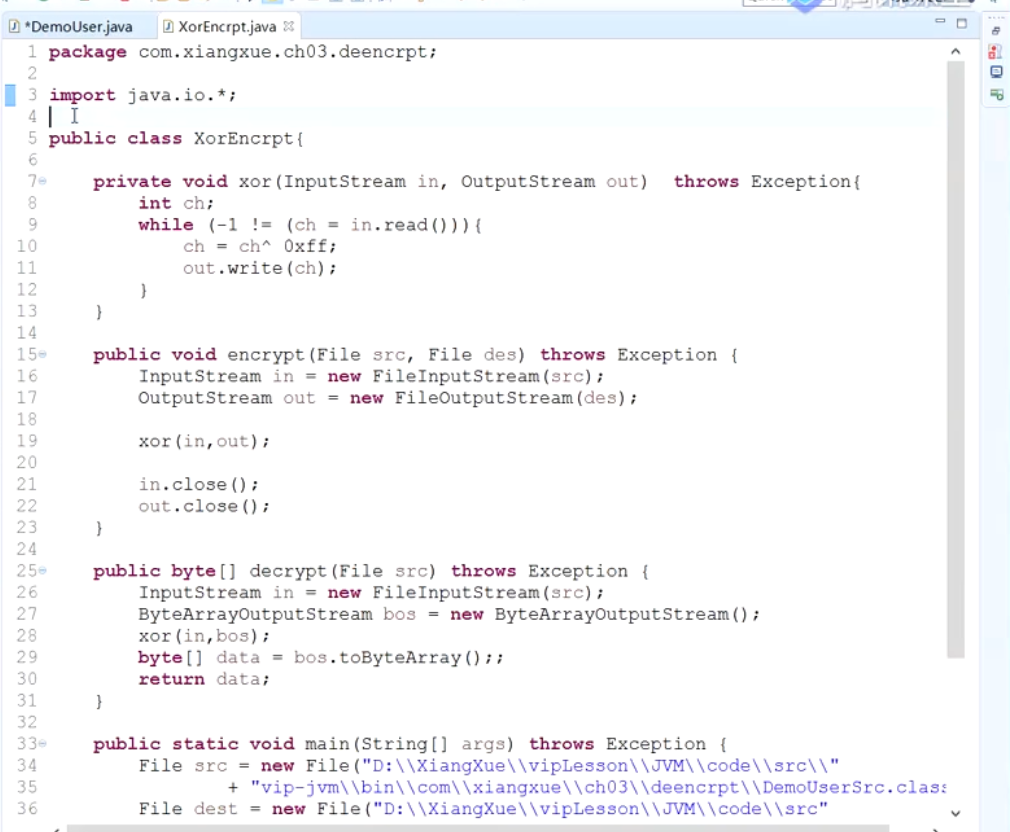
准备阶段做内存分配，比如static int 设置为0

初始化阶段执行类的构造器 <clinit>，初始化过程，按照源码流程，如果没有静态属性，编译器不会生成<clinit>

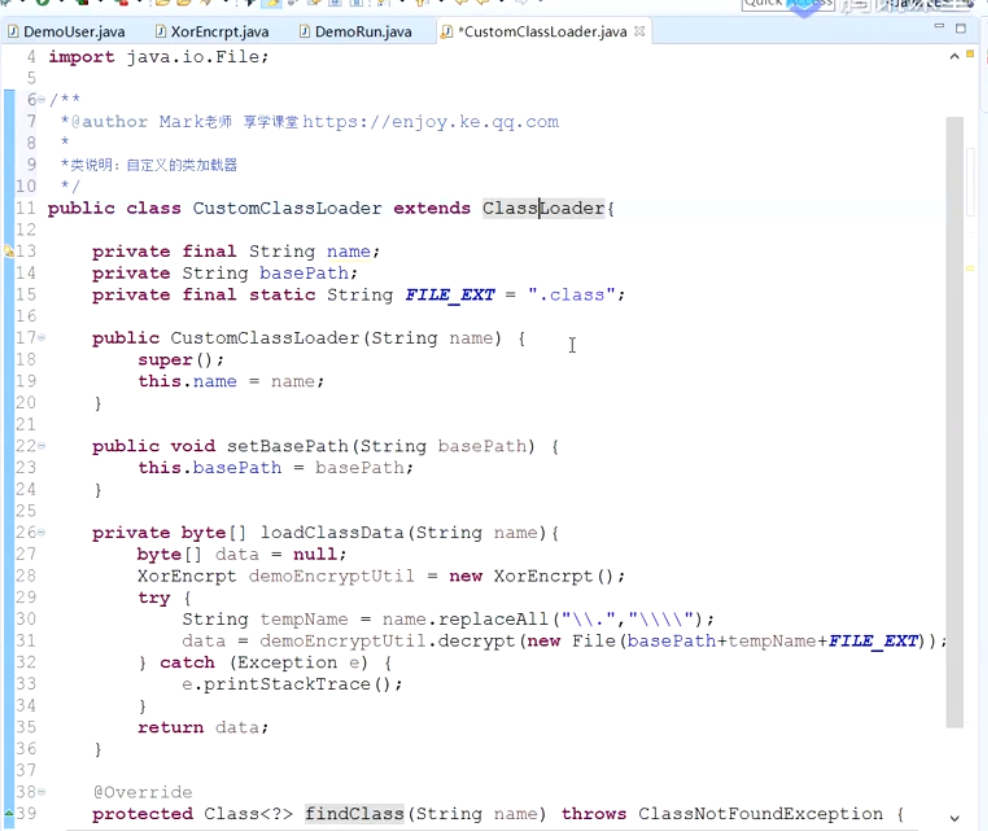




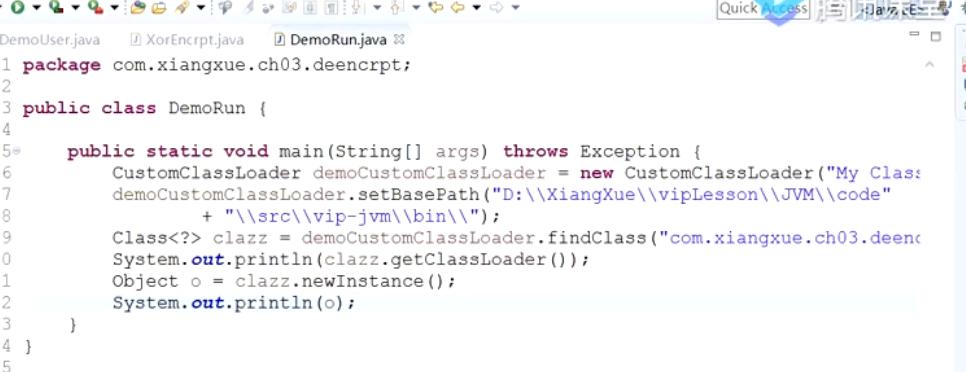


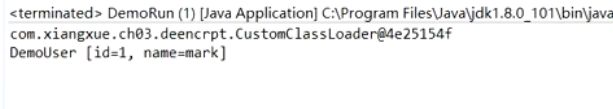




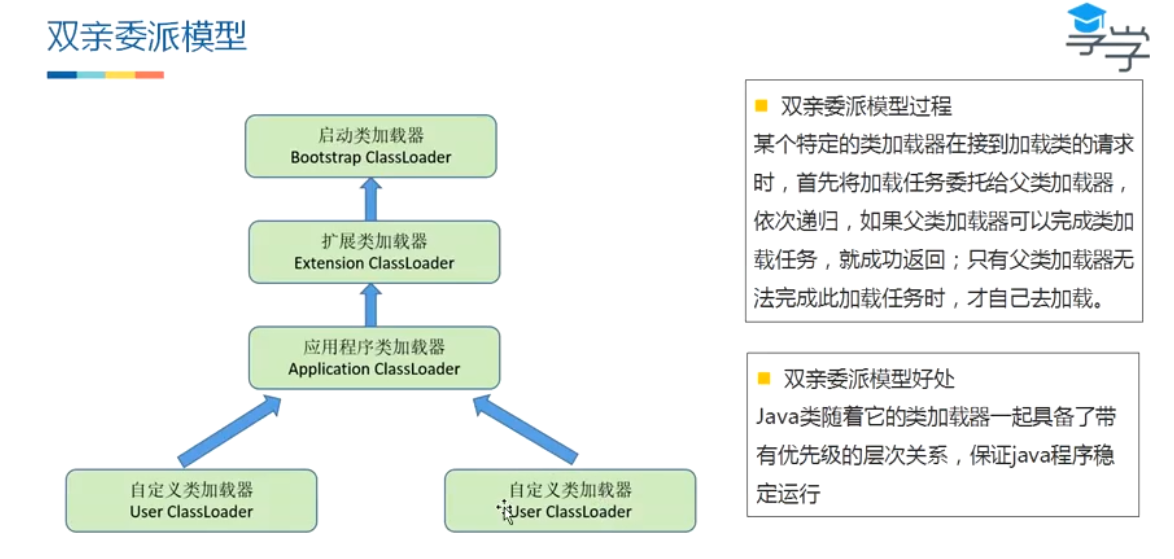




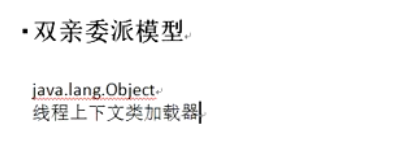




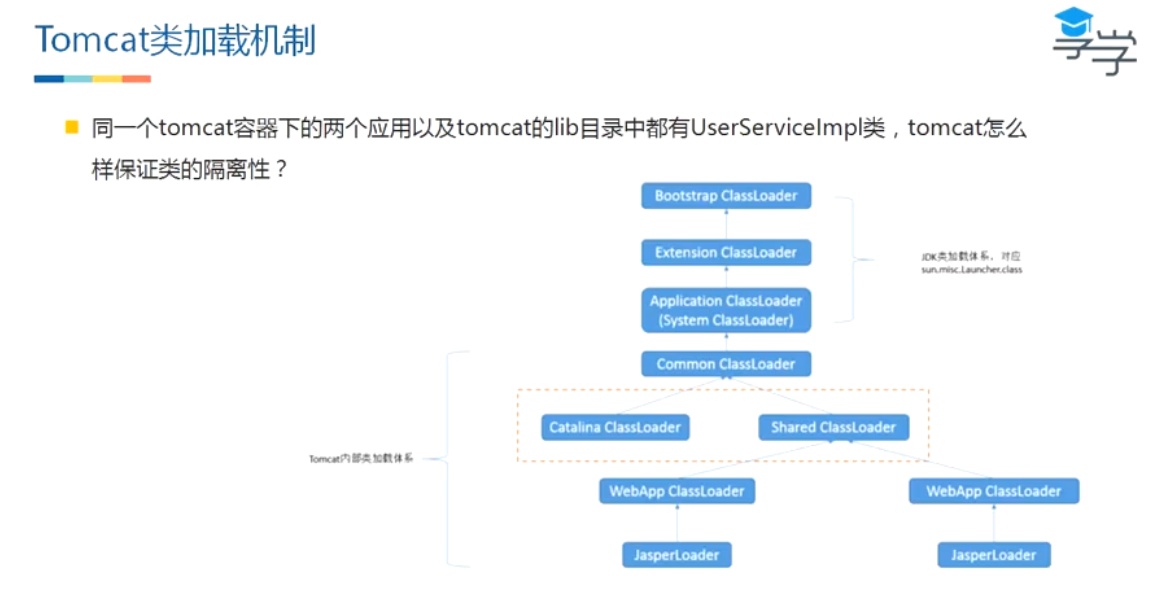


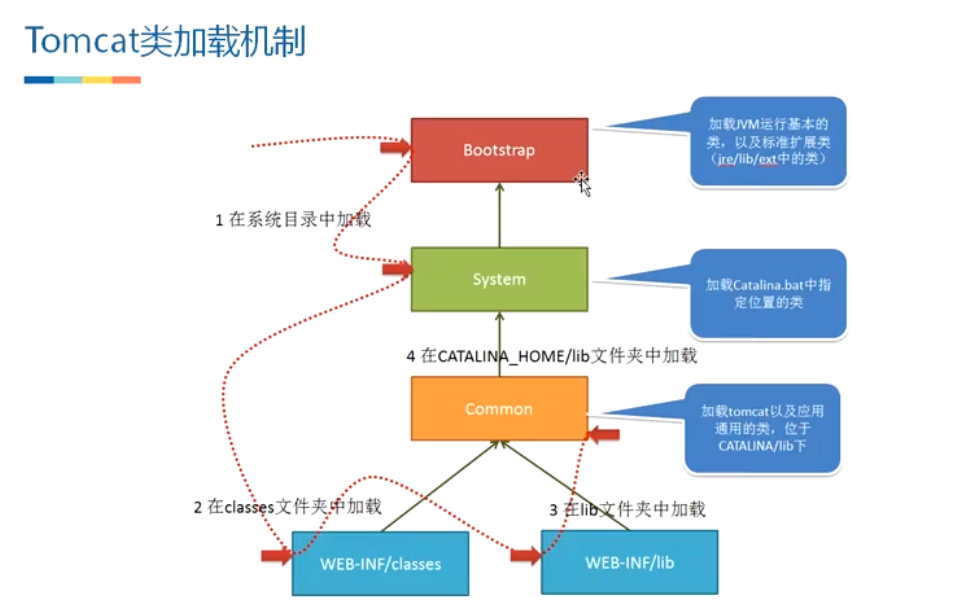


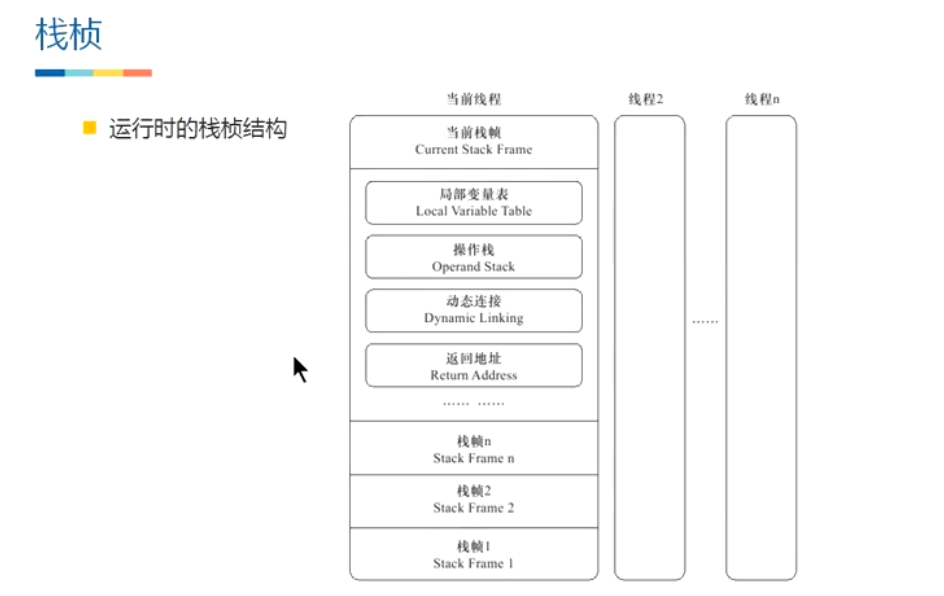
委托复合关系，不是继承关系



Java中怎么破坏双亲委派？



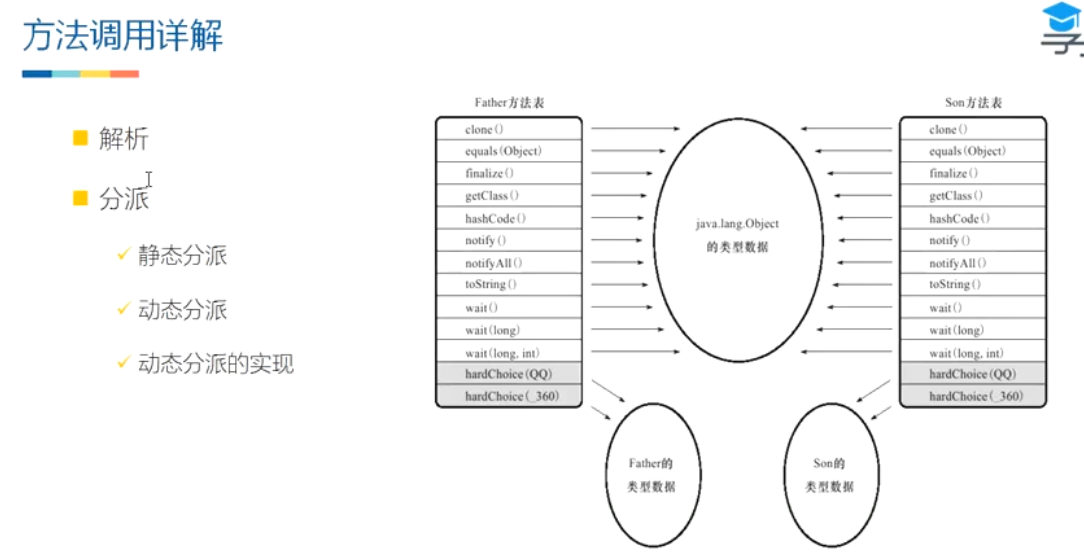




局部变量表：不是方法定义了多少局部变量，局部变量表中就有多少个局部变量，可能会卸载。



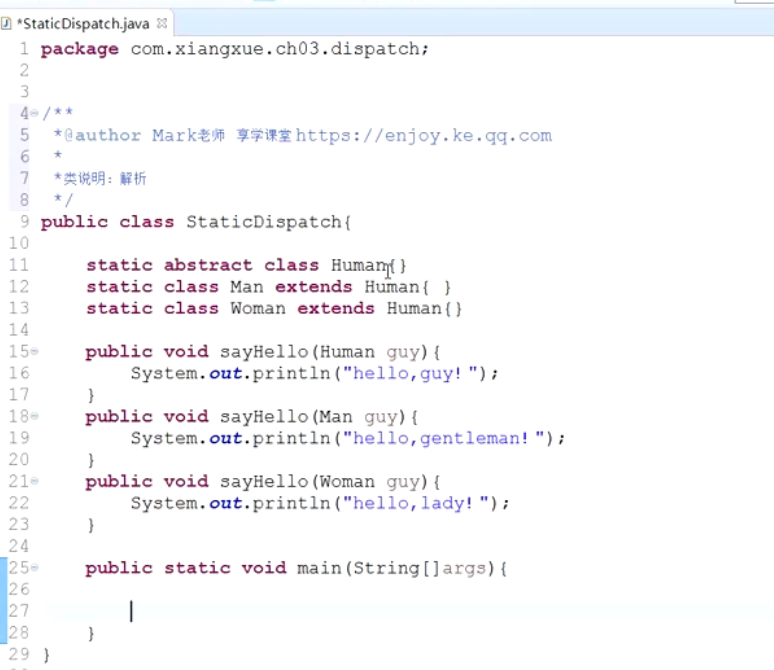
栈帧 是一个抽象概念，包含了很多东西，不仅仅包含一块内存，可能还有CPU高速缓存，具体由虚拟机决定。

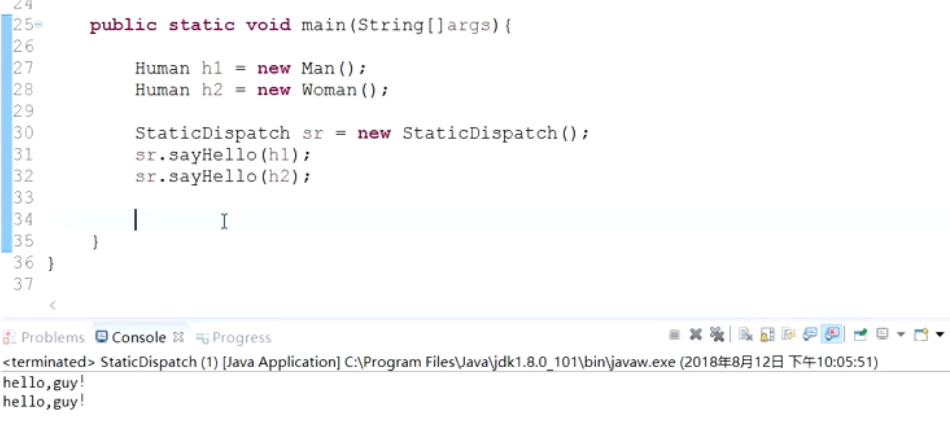


解析是什么？

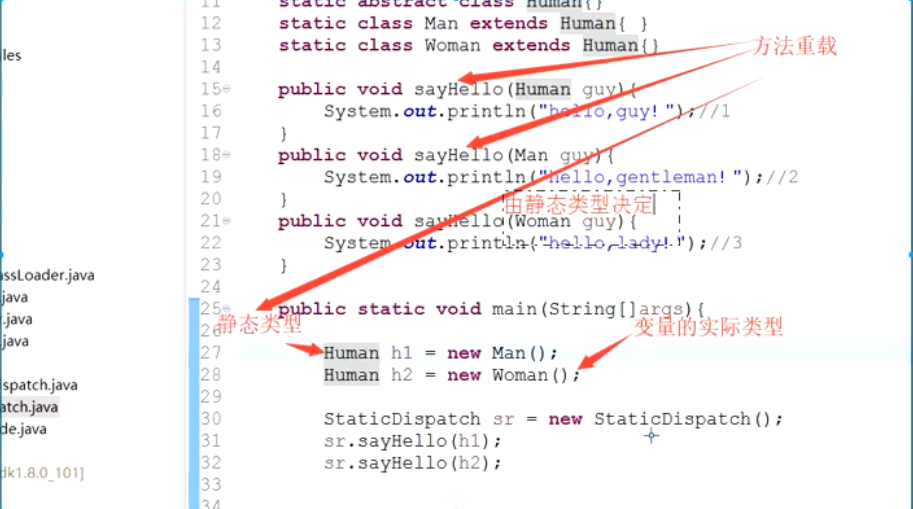
类的静态方法，构造方法，私有方法的调用时，通过解析指定。

分派，静态解析



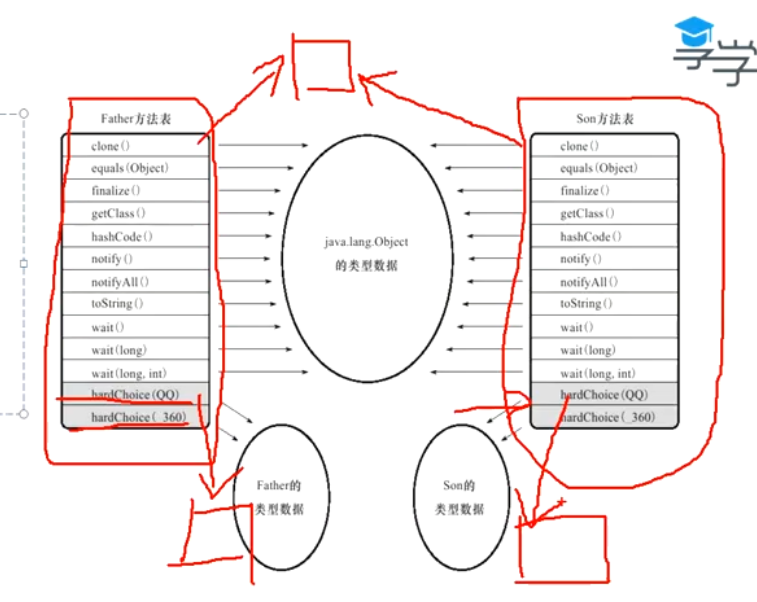


结果解析：



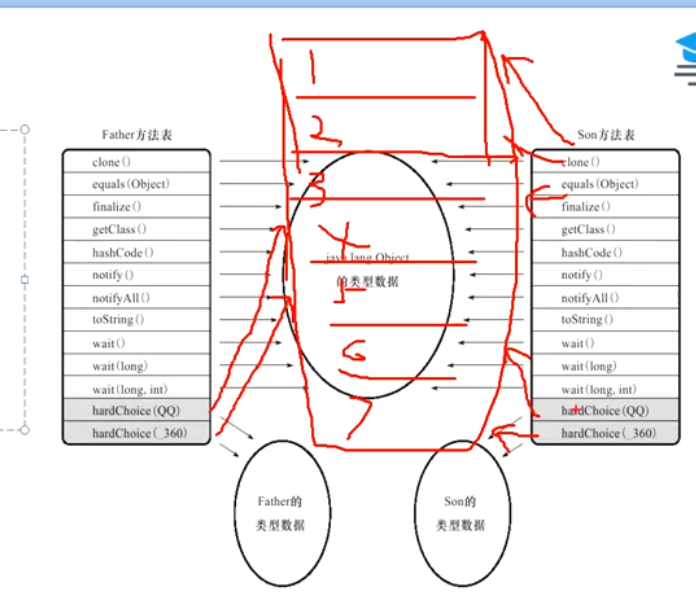
Java 方法重载调用时，是根据静态类型来匹配的，这就是方法调用中的， 分派----------静态分派

分派---------------动态分派 方法的重写，调用时涉及到动态分派。



这就是方法重载，多态。

动态分派时，存在虚方法表，虚拟机会根据如下方法表，找到到对应的方法地址



虚方法表：虚拟机维护，维护各个方法的实际入口，就是方法区中的各个内存地址。

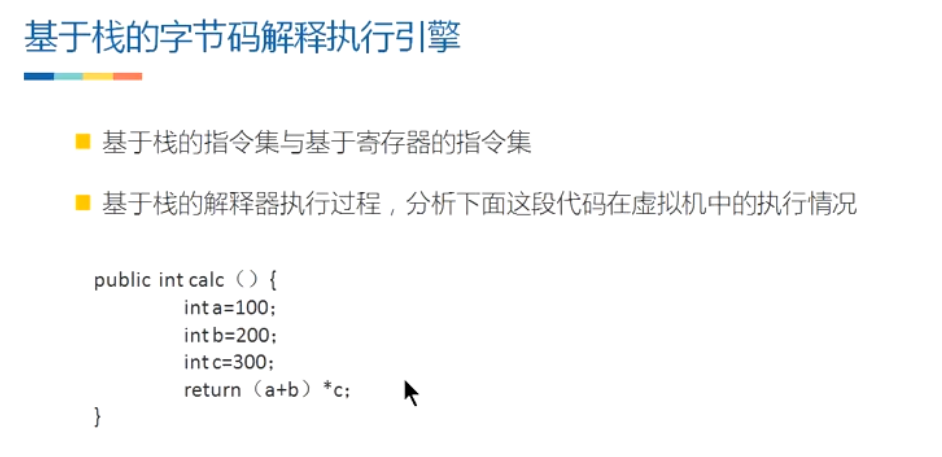
一个类，一个虚方法表

对象的多态，就是由动态分派决定。

解析、静态分派是编译期就决定了。

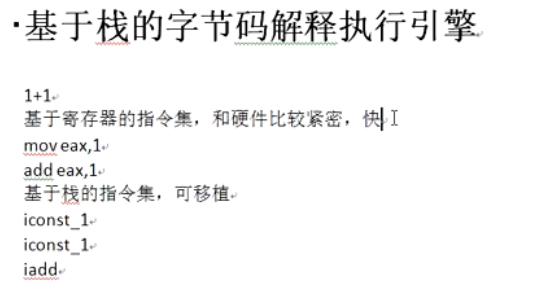
课程总结：

双亲委派破坏，比如JDBC加载

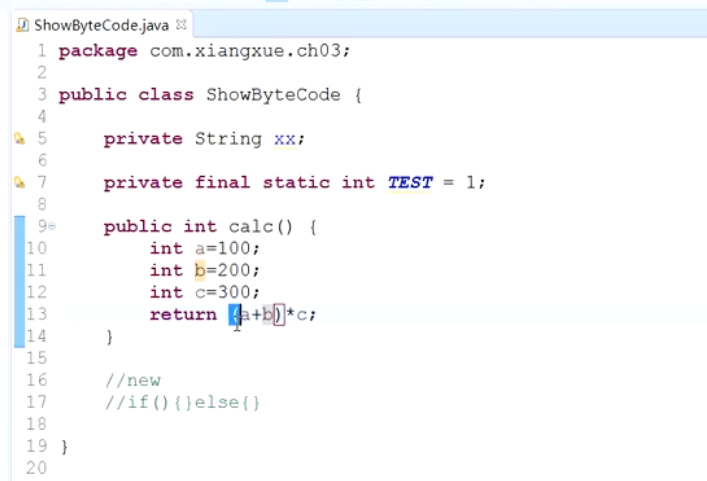


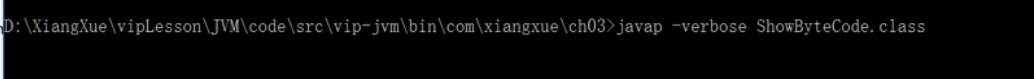
基于栈：一个指令执行时，指令的大部分都是没有地址的，需要依赖操作数栈来执行。

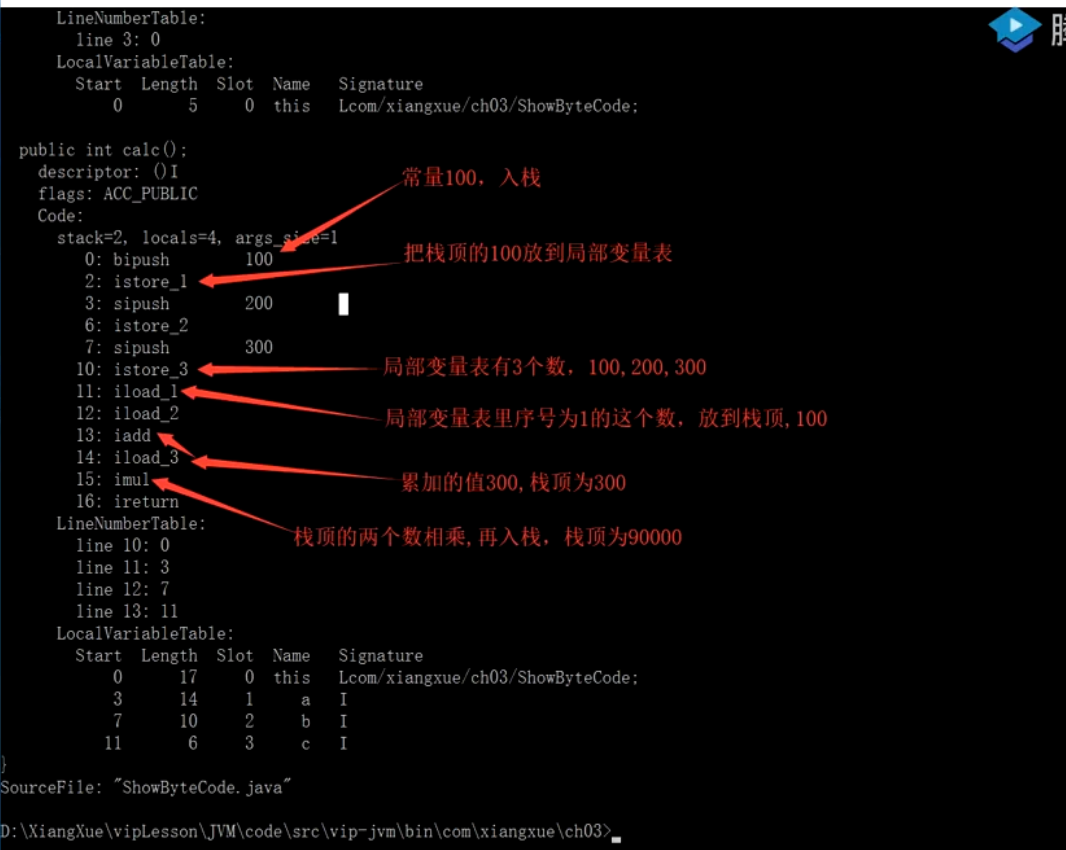
基于寄存器：window x86 就是基于集群器的指令集



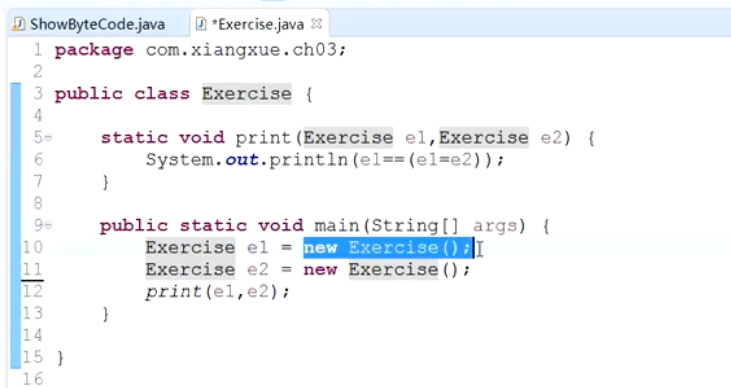
代码实例：







作业：



分析该代码：通过字节码解释为什么打印false