**>>类文件结构**

1.write once, run anywhere!

2.Java虚拟机并不仅仅只是运行Java语言，还可以支持其他语言，其他语言使用对应的编译器将源代码编译成字节码文件（即.class文件）

3.class文件解析

1）class前四个字节称为Magic Number，作用是确定这个class文件是否能被虚拟机接受，因为后缀名是可以修改的。

2）接下来的四个字节是Class文件的版本号，第5，6个字节是Minor Version，第七八个字节是Major Version.

>>虚拟机加载机制

1. 类的生命周期：
   1. 加载->验证->准备->解析->初始化->使用->卸载
2. 对类进行初始化有且仅有五中情况
   1. 遇到new, getstatic, setstatic, invokestatic四条字节码指令，如果类没有过初始化，则先触发初始化
   2. 使用java.lang.reflect包的方法进行反射调用的时候，如果类没有进行过初始化先触发其初始化
   3. 当初始化类的时候，如果父类没有初始化，则先初始化父类；一个接口在初始化的时候，并不要求其父类接口都完成初始化，只是真正使用到父接口的时候才会初始化
   4. 虚拟机启动时，包含main方法的类会初始化
   5. 当使用JDK1.7的动态语言支持时，如果一个java.lang.invoke.MethodHandle实例最后的解析结果REF\_getStatic..的方法句柄，并且对应类没有初始化，则需要先触发器初始化。
   6. 通过子类引用父类中的静态字段不会导致子类初始化
   7. 调用类中的常量也不会触发类的初始化
3. 类加载过程：加载，验证，准备，解析，初始化
   1. 加载过程需要完成三件事：
      1. 通过类的全限定名获取定义此类的二进制文件
      2. 将这个字节流所代表的静态存储结构转化为方法区运行时的数据结构
      3. 在内存中生成java.lang.Class对象，作为方法区各个数据访问入口
   2. 加载和连接是交叉进行的，但是加载和连接的交叉顺序任然是加载在前
   3. 验证是连接的第一步，确保Class文件中的信息符合虚拟机的要求，不会危害虚拟机自身的安全。
   4. 验证大概分为以下四部分：
      1. 验证文件格式：根据文件中的各个位置的十六进制数判断是否符合规范，如魔数0XCAFEBABE开头，主次版本号等等
      2. 元数据验证：对类的元数据信息进行语意校验
      3. 字节码验证：主要是对方法体中的一些验证，包括局部变量的类型，是否会运行到方法体之外的代码等等。
      4. 符号引用验证：确保解析动作能够正常执行
   5. 准备过程：为类变量（不包括实例变量,实例变量在java堆中）分配内存和设置类变量初始值（为默认值0，而不是”=”之后的值）的阶段，内存在方法去中分配。
   6. 解析过程