设计之初，设计者们就曾想过让Java以外的语言（Scala， JRuby， Groovy）同样生成.class文件， JVM只关心.class文件， 翻译成字节码文件， 而不关心语言本身。 从侧面也说明JVM的字节码命令功能要超过java语言本身很多。
JVM主要描述了class文件结构 和 字节码指令集。

6.1 class文件的结构

任何一个.class文件都应对着一个类或接口的定义信息。但是反过来， 类或接口可以不用定义在class文件中， 比如可以直接通过类加载器直接生成。
class文件， 以一组以8bite为单位的二进制流， 其只包括 无符号数（u1，u2,u4,u8代表大小哈） 和 表（ JVM由多个无符号数和其他表构成构成） 两种结构。
1）魔数 与 版本号
Magic Number， 4个字节， 作用为： 确认这个文件是否是能被JVM接受的.class文件。在其他领域还可以用魔数来作为文件的格式（扩展名易变）比如 JPEG
2个字节的 次版本号（Minor Version） 和 2个字节的 主版本号（Major Version），能往前兼容， 不能往后， 最新的1.7版本号是 51，最早的是45

2） 常量池
class文件资源的仓库， 大小不定， 所以由第一个counter计数， 主要有 字面量（类似Java中的常量概念）和符号引用（类和接口的全限定名， 字段的名称和描述符， 方法的描述符和名称）。当JVM运行时， 需要从此处获得对应的符号引用， 再在类创建或者运行时解析， 翻译到具体的地址中

3）访问标志
2个字节，用于标识一些类或者接口的访问信息： 这个类是类还是接口？是否定成public？ 是否为abstract？有没有final？。。。。

4）类索引， 父类索引，与接口索引的集合。
前两个都是2个字节， 最后一个是一组2B数据的集合。主要描述类的继承关系， 哪些类实现了接口。。。

5）字段表集合
描述类和接口中声明的变量。 包括类中的static变量， 实例级变量， 但是不包括方法内部声明的局部变量。

6）方法表集合

包括了 访问标志， 名称索引， 描述符索引， 属性表集合。一个方法的定义可以通过前面几个确定， 名字可以放在常量池， 但是方法里面的代码却放在了方法属性集合中一个名为"code"属性的集合中。

7）属性表集合
包括6）中提到的code集合（用于描述方法代码）， Exception属性（列举出方法中可能抛出来的异常）， LineNumberTable（Java源代码和字节码行号的对应关系）。。。。。。。。。

6.2 字节码指令集

包括了加载与存储指令， 运算类指令， 类型转化指令， 对象创建与访问指令（newarray，getfield,baload ,bastore,arraylength,instanceof....）， 操作数栈指令，控制转移类指令， 方法调用与返回， 异常处理， 同步指令（使用管程）。

JVM实现的class文件结构和字节码指令集的内容与硬件， 操作系统是完全独立的， 从而实现跨平台的手段