/\*\*

\* java class文件结构

\*

\* 序言：

\*

\* 代码编译的结果从本地机器代码转变为字节码，是存储格式发展的一小步，编程语言发展的一大步

\*

\* class文件是java平台独立、语言独立的基础

\*

\* class文件是由字节码指令组成，按照严格的格式描述了用各种编程语言书写的代码的语义的中间结果

\*

\* 字节码提供的语义描述能力远比java的语义描述能力强大

\* 正因为此，java无法提供的特性并不一定其他运行在JVM上的语言也无法提供

\* 目前能够运行在JVM上的语言有很多，他们或多或少的提供了一些java没有的特性，比如代码简洁、支持函数式编程等等

\* 其他JVM-BASED语言：JRuby、Groovy、JPython、Scala、Clojure

\*

\* JVM为这些语言提供的了一些额外的支持，是的这些语言更为容易在JVM上运行，比如：

\* 1、JDK1.5开始的，脚本引擎的加入

\* 2、JDK7中动态调用指令的添加

\*

\* class文件是由二进制字节码组成，最基本的单位是8位字节。各个数据项严格的按照一定顺序排列

\* class文件的字节序采用的是big-endian,即大端（高位在前的方式排列）

\* class文件的各个数据项紧密的排列在一起，没有对齐及填充因子这些东西（性能换空间，牺牲性能，换取空间，Java是为网络而设计的）

\*

\* class文件采用类似c语言结构体的格式来描述，只有两种数据类型。无符号数与表

\*

\* 无符号数：用来描述数值、索引项、数量值或是以UTF-8编码的字符串值。u1、u2、u4、u8

\* 表： 由多个无符号数或是其他表组成的，用来描述有层次关系的复合数据结构。class文件本身就是一种表结构

\*

\* class文件中每个条目都称为一个数据项，整个文件就是由这些数据项组成

\*

\* 多个同一类型的数据项是由表来组织的，这个表的前一个数据项会记录这个数据项包含的数据项数量

\*

\* 常量池：

\* class文件中占用存储空间比较大的数据项之一。

\* 主要用于存储以下内容：

\* 1、字面量，包括文本字符串、被申明为final类型的常量等

\* 2、符号引用：类或是接口的全限定名称、方法以及字段的名称及描述符

\*

\* 第一个的原因估计是想这样数据结构来减小class文件的大小

\* 第二个主要是编译原理中连接器有关，java class文件结构并不能描述方法、字段的最终内存布局

\* 而是在类加载器加载后进行解析或是运行时才把方法、字段解析到具体的内存地址上的取得

\*

\* 常量池是一个表结构，它的数据项也都是表类型数据项

\* 总共有十一种表类型数据项，这十一哥数据项又一个共同点就是第一位是一个u1类型的标志位，标识这个数据项的代表的常量的类型

\* 这十一个表项的数据结构各部相同

\*

\*

\* 访问标志位

\* 用于识别类或接口的访问信息，标识是一个类、接口、枚举还是注解，是否为public,是否为abstract,是否为final类型等等

\* 这部分总共可以使用32位，目前使用8位，其他的均置为0

\*

\* 类索引、父类索引、接口索引

\* 类索引、父类索引均是u2类型数据项，指向常量池中类及父类的全限定名称项

\* 接口索引是由u2类型的数据项集合组成，接口索引的第一项记录接口的数量

\*

\* 字段表

\* 用于描述接口及类中定义的字段，包括类级、实例级（不包括方法中定义的局部变量）

\* 主要包括以下信息：

\* 1、字段的作用域 2、类级还是实例级 3、可变性 4、并发性 5、字段类型 6、字段名称 7、是否可序列化8、字段描述符

\*

\* 名词解释

\* 全限定名称：就是包含包名及java文件的名称，如com/ares/Test.java

\* 简单名称：就是Test.java

\* 描述符：这个相对全限定名称与简单名称要复杂多了，他用于描述字段的类型、方法的参数列表及返回值

\* 基本数据类型都用一个大写字母表示，对象用L加全限定名称来表示，对于数组每一维度使用一个 [ 来表示

\*

\* 方法表

\* 方法表用于描述接口或是类中定义的方法，结构上与字段表基本一致，但是描述的信息是用于说明方法的

\* 主要包括以下信息：

\* 1、作用域2、是否为类方法3、是否可覆盖4、是否为同步方法5、是否为本地方法6、是否为抽象方法

\* 7、是否为由编译器产生的桥接方法（Enum就是例子）8、是否由编译器自动产生9、是否为strictfp 10、方法描述符

\* 方法体是存放在方法表中属性集合中一个叫做CODE的属性里面

\*

\* 属性表集合

\* 字段表与方法表都有一个叫做属性表的数据项，它是用于描述某些特殊场景下的信息

\* HotSpot中预定义了如下属性信息：

\* 1、Code 用于方法表，存放方法体编译后的字节码指令

\* 2、ConstantValue,用于字段表，描述常量值

\* 3、Deprecated,用于类、方法表及字段表，描述被废弃的类、方法及字段

\* 4、Exceptions,用于方法表，描述方法抛出的异常

\* 5、InnerClasses,用于类文件，描述内部类列表

\* 6、LineNumberTable,用于Code属性，描述了java源代码行号与字节码指令的映射

\* 7、LocalVarialTable,用于Code属性，描述了方法的参数列表

\* 8、SourceFile,用于类文件，描述了方法源文件名称

\* 9、Synthetic,用于类、方法表及字段表，描述了是否有编译器生成的

\*

\* Code属性

\* Code属性用于存放方法体编译后的字节码指令，但并不是所有方法都有这个属性，比如抽象方法、接口中的方法是可以没有Code属性的

\*

\*/