Class文件的结构（这是class文件的所有内容）



ClassFile 结构体中，各项的含义描述如下：

 magic

魔数，魔数的唯一作用是确定这个文件是否为一个能被虚拟机所接受的 Class 文件。魔

数值固定为 0xCAFEBABE，不会改变。

 minor\_version、major\_version

副版本号和主版本号， minor\_version 和 major\_version 的值分别表示 Class 文件的副、 主版本。 它们共同构成了 Class 文件的格式版本号。 一个Java虚拟机只支持特定的版本号的字节码

 constant\_pool\_count

常量池计数器， constant\_pool\_count 的值等于 constant\_pool 表中的成员数加 1。

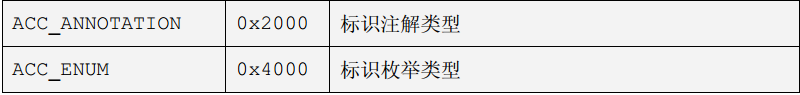
 constant\_pool[]

常量池，constant\_pool 是一种表结构（§4.4） ，它包含 Class 文件结构及其子结构中引用的所有字符串常量、 类或接口名、字段名和其它常量。 常量池中的每一项都具备相同的格式特征——第一个字节作为类型标记用于识别该项是哪种类型的常量，称为 “tag byte” 。

 access\_flags

访问标志，access\_flags 是一种掩码标志，用于表示某个类或者接口的访问权限及基础属性。





 this\_class类索引，this\_class 的值必须是对 constant\_pool 表中项目的一个有效索引值。

constant\_pool 表在这个索引处的项必须为 CONSTANT\_Class\_info 类型常量，表示这个 Class 文件所定义的类或接口。

 super\_class

父类索引，对于类来说，super\_class 的值必须为 0 或者是对 constant\_pool 表中项目的一个有效索引值。如果它的值不为 0，那 constant\_pool 表在这个索引处的项必须为 CONSTANT\_Class\_info 类型常量

 interfaces\_count

接口计数器，interfaces\_count 的值表示当前类或接口的直接父接口数量。

 interfaces[]

接口表，interfaces[]数组中的每个成员的值必须是一个对 constant\_pool 表中项目的一个有效索引值， 它的长度为 interfaces\_count。

 fields\_count

字段计数器，fields\_count 的值表示当前 Class 文件 fields[]数组的成员个数。

 fields[]

字段表，fields[]数组中的每个成员都必须是一个 fields\_info 结构（§4.5）的数据项，用于表示当前类或接口中某个字段的完整描述。

 methods\_count

方法计数器， methods\_count 的值表示当前 Class 文件 methods[]数组的成员个数。

 methods[]

方法表，methods[]数组中的每个成员都必须是一个 method\_info 结构（§4.6）的数据项，用于表示当前类或接口中某个方法的完整描述。

 attributes\_count

属性计数器， attributes\_count 的值表示当前 Class 文件 attributes 表的成员个数。

 attributes[]

属性表，attributes 表的每个项的值必须是 attribute\_info 结构（§4.7）。在本规范里，Class 文件结构中的 attributes 表的项包括下列定义的属性：InnerClasses 、 EnclosingMethod 、Synthetic、Signatur、SourceFile，SourceDebugExtension 、Deprecated、RuntimeVisibleAnnotations、RuntimeInvisibleAnnotations以及BootstrapMethods属性。

常量池的结构：

Java 虚拟机指令执行时不依赖与类、接口，实例或数组的运行时布局，而是依赖常量池

（constant\_pool）表中的符号信息。常量池有多个cp-info组成，而cp\_info具有如下格式

cp\_info {

u1 tag;

u1 info[];

}

方法

所有方法（Method）， 包括实例初始化方法和类初始化方法在内， 都由 method\_info

结构所定义。在一个 Class 文件中，不会有两个方法同时具有相同的方法名和描述符

method\_info 结构格式如下：

method\_info {

u2 access\_flags; 访问权限控制（public/private）

u2 name\_index;

u2 descriptor\_index;

u2 attributes\_count;

attribute\_info attributes[attributes\_count];

}

属性

属性（Attributes）在 Class 文件格式中的 ClassFile结构、 field\_info结构，method\_info结构和 Code\_attribute结构都有使用，所

有属性的通用格式如下：

attribute\_info {

u2 attribute\_name\_index;

u4 attribute\_length;

u1 info[attribute\_length];

}

Code 属性

Code 属性是一个变长属性，位于 method\_info（§4.6）结构的属性表。一个 Code 属性

只为唯一一个方法、实例类初始化方法或类初始化方法（§2.9）保存 Java 虚拟机指令及相关辅助信息。 所有 Java 虚拟机实现都必须能够识别 Code 属性。如果方法被声明为 native 或者abstract 类型，那么对应的 method\_info 结构不能有明确的 Code 属性



在code属性中有个栈映射帧的属性，它们与运行时栈帧中的操作数栈、局部变量表有映射关系，但并非同一样东西。

LocalVariableTable 属性

LocalVariableTable 是可选变长属性，位于 Code（§4.7.3）属性的属性表中。它被调试器用于确定方法在执行过程中局部变量的信息。于此同时还有字节码与源代码中代码的对应关系

RuntimeVisibleAnnotations 属性

RuntimeVisibleAnnotations 属性是可变长属性，位于 ClassFile（§4.1），field\_info（§4.5）或 method\_info（§4.6）结构的属性表中。RuntimeVisibleAnnotations 用于保存 Java 语言中的类、字段或方法的运行时的可见注解（Annotations）。