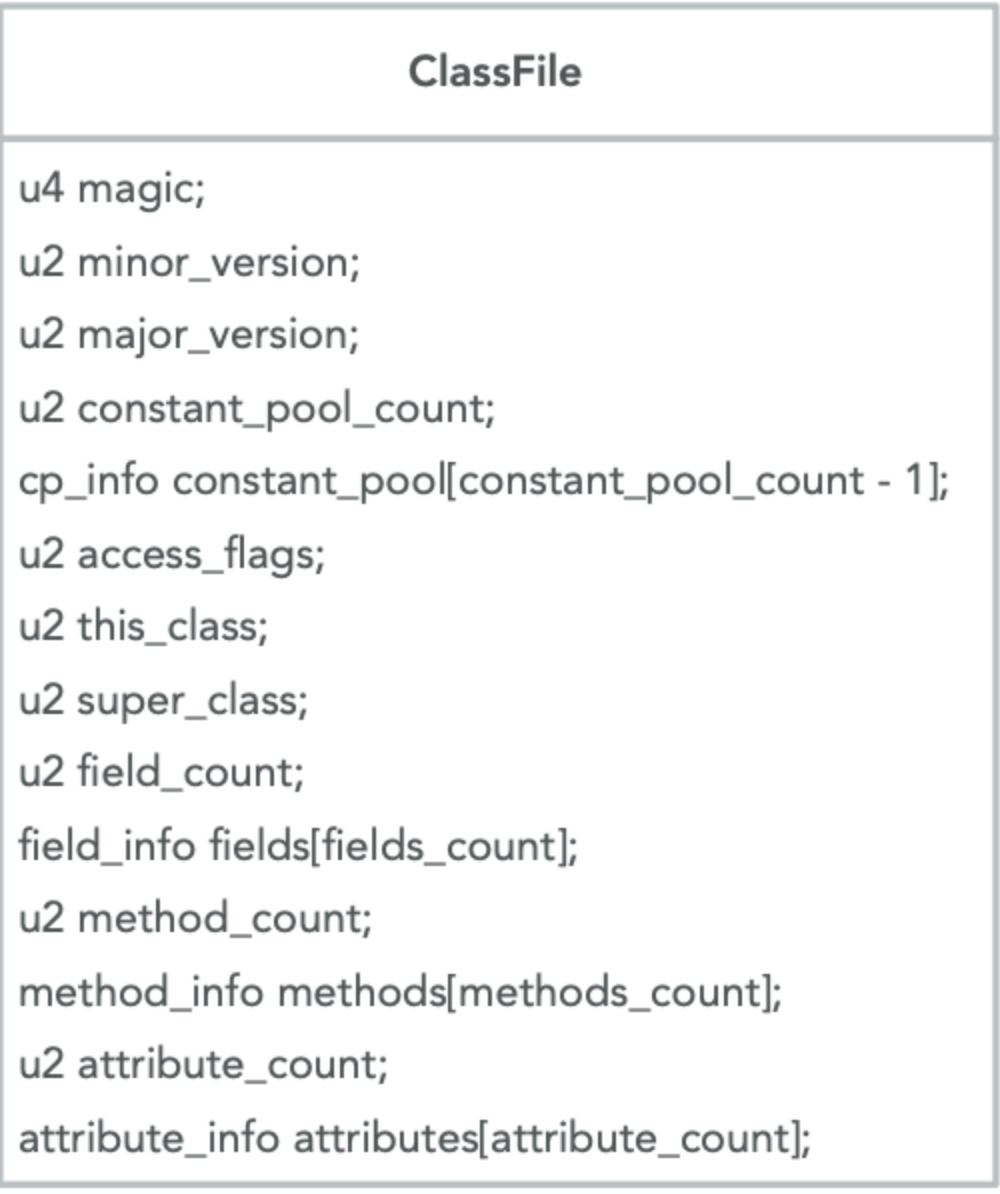
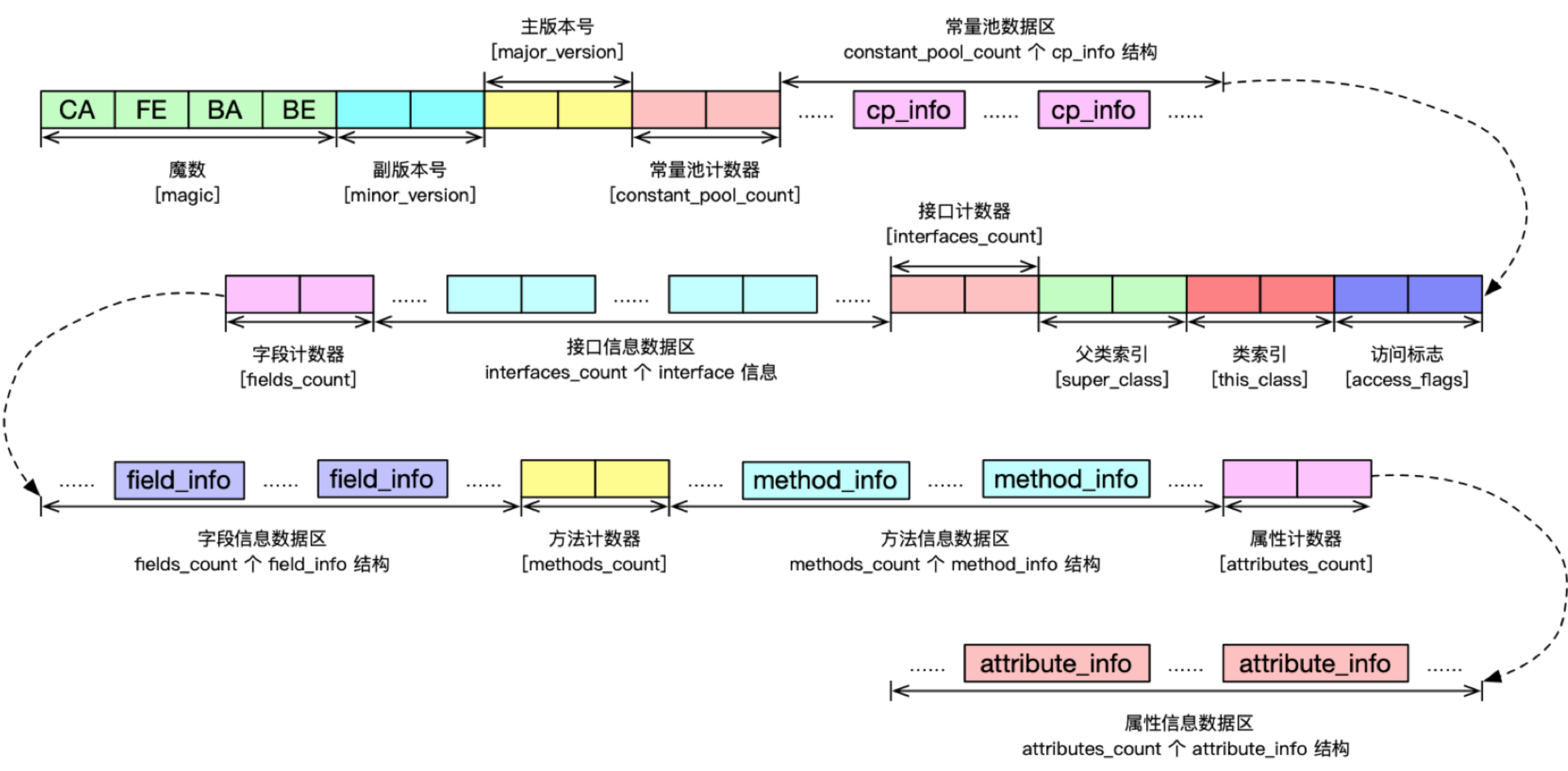
# **Class文件的基本结构**

我们知道，编写的 .java 源文件会被 java 编译器编译成 .class 文件，它是由字节组成的文件，又叫字节码文件。

那你知道 class 文件里面都存储了哪些数据吗？这里我绘制了 class 文件的 UML 图和基本组成结构。





## **魔数**

所有 java 编译器编译的 .class 文件的前4个字节都是 0xCAFEBABE，被称为魔数(magic)。

它的作用在于，JVM 在尝试加载某个文件的时候，会先读取文件的前4个字节，判断是否是 0xCAFEBABE；如果是，则 JVM 会将此文件作为 class 文件来加载并使用。

## **版本号**

版本号表示当前 class 文件被哪个版本的 java 编译器编译而成。

1. 副版本号(minor\_version)，占用第 5、6 两个字节；
2. 主版本号(major\_version)，占用第 7、8 两个字节。

JDK1.0 的主版本号(major\_version) 为 45，以后的每个新主版本(major\_version)都会在原先版本的基础上加1。

JVM 在加载 class 文件的时候，会读取其主版本号(major\_version)，然后比较主版本号(major\_version) 和 JVM 本身的版本号，如果 JVM 本身的版本号小于 class 文件的主版本号(major\_version)，JVM 会认为加载不了这个 class 文件，并抛出 java.lang.UnsupportedClassVersionError。

举个栗子，这里我用 java11 编译生成 .class 文件，然后使用 java8 运行。



## **常量池计数器**

常量池计数器(constant\_pool\_count)描述着整个 class 文件的字面量数量。

## **常量池数据区**

常量池(constant\_pool)是由若干 cp\_info 的结构体组成。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(constant\_pool)数组长度由常量池计数器(constant\_pool\_count)指定，它们之间满足如下关系。

// 常量池计数器 = 常量池数组长度 + 1

constant\_pool\_count = constant\_pool.length + 1

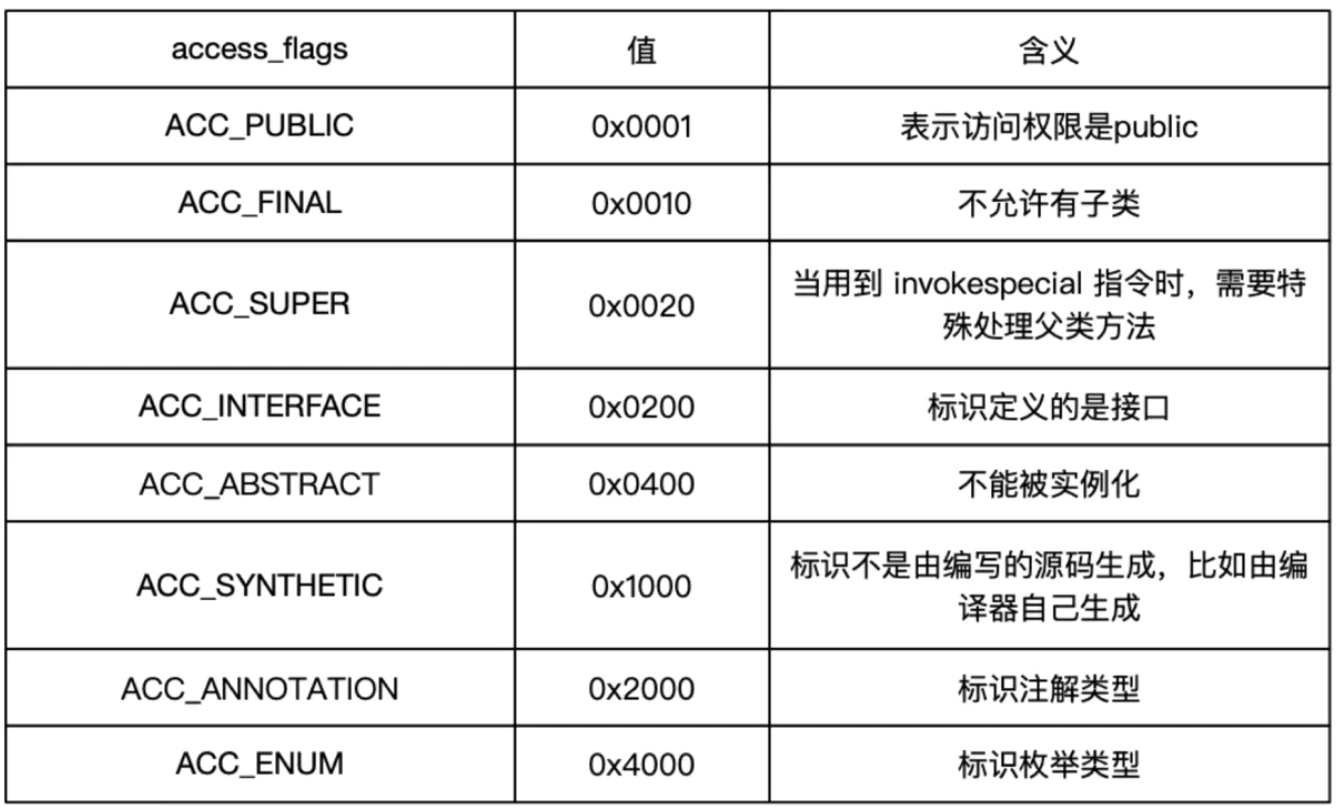
常量池(constant\_pool)数组的索引范围是 [1, constant\_pool\_count − 1]，这里的索引不是从 0 开始的。

将索引为 0 的常量池项空出来是为了满足特定情况下指向常量池索引的数据不引用任何一个常量池项，这种情况下可以将常量池索引设置为0，比如 java.lang.Object 的父类索引(super\_class)。

## **访问标志**

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。

一些常见的值以及其含义如下图所示。



ACC\_INTERFACE 表示是接口。如果一个 class 文件有 ACC\_INTERFACE 标志，那么也得设置 ACC\_ABSTRACT；并且不能设置为 ACC\_FINAL、ACC\_SUPER、ACC\_ENUM；

注解类型必须带有 ACC\_ANNOTATION 标志，如果设置了 ACC\_ANNOTATION 标志，ACC\_INTERFACE 也必须同时被设置；

ACC\_SUPER 标志用于确定该 class 文件里面的 invokespecial 指令使用的是哪一种执行语义。目前 java 编译器应该都会设置这个标志，ACC\_SUPER 是为了兼容旧编译器编译 class 文件而存在的；

## **类索引**

类索引(this\_class)表示这个 class 文件所定义的类或接口。

类索引(this\_class) 的值必须是常量池(constant\_pool) 数组中的一个有效索引值，且常量池(constant\_pool)数组在此索引处的项必须是 CONSTANT\_Class\_info 类型的。

## **父类索引**

父类索引(super\_class)表示这个 class 文件所定义类或接口的直接父类。

对于非 java.lang.Object 的类，父类索引(super\_class)的值必须是常量池(constant\_pool)数组中的一个有效索引值，且常量池(constant\_pool)数组在此索引处的项必须为 CONSTANT\_Class\_info 结构；

对于 java.lang.Object 类，父类索引(super\_class)的值一定是 0；

对于接口，父类索引(super\_class)的值必须是常量池(constant\_pool)数组中的一个有效索引值，且常量池(constant\_pool)数组在此索引处的项必须是代表 java.lang.Object 类的 CONSTANT\_Class\_info 结构；

类的直接父类以及所有间接父类的访问标志(access\_flag)中都不能带有 ACC\_FINAL 标记。

## **接口计数器**

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

## **接口信息数据区**

接口信息(interfaces)是一个索引数组，表示当前类或接口的直接父接口信息。

接口信息(interfaces)数组的长度为接口计数器(interfaces\_count)的值。数组中每一项必须是常量池(constant\_pool)数组中的一个有效索引值，且常量池(constant\_pool)数组在此索引处的项必须为 CONSTANT\_Class\_info 结构。

数组表示的接口顺序和对应的源文件中给定的接口顺序(从左至右)一样，即第一个元素对应的是源文件中最左边的接口。

## **字段计数器**

字段计数器(fields\_count)表示当前 class 文件声明的所有字段数量，不包括从父类或父接口继承的字段。

## **字段信息数据区**

字段表(fields)表示当前 class 文件声明的所有字段，不包括从父类或父接口继承的方法。

字段表(fiels)的长度为字段计数器(field\_count)的值。数组中每一项都必须是 field\_info 结构，表示当前 class 文件中某个字段的完整描述。

## **方法计数器**

方法计数器(methods\_count)表示当前 class 文件声明的方法数量，不包括从父类或父接口继承的方法。

## **方法信息数据区**

方法表(methods)表示当前 class 文件声明的所有方法，不包括从父类或父接口继承的字段。

方法表(methods)的长度为方法计数器(method\_count)的值。数组中每一项都必须是 method\_info 结构，表示当前 class 文件中某个方法的完整描述。

如果方法表(methods)中某一项的 access\_flags 既没有设置 ACC\_NATIVE 标志，也没有设置 ACC\_ABSTRACT 标志，那么它所对应的方法体就应当可以被 JVM 直接从当前类加载，而不需要引用其它类。

## **属性计数器**

属性计数器(attributes\_count)表示当前 class 文件的所有属性数量。

## **属性信息数据区**

属性表(attributes)表示当前 class 文件的所有属性。

属性表(attributes)的长度为属性计数器(attribute\_count)的值。表中每一项都必须是 attribute\_info 结构，描述某个属性的完整信息。

Java 7 规范里，class 文件结构中的属性表(attributes)可以包括下列定义的属性：InnerClasses 、 EnclosingMethod、Synthetic、Signature、SourceFile、SourceDebugExtension、Deprecated、RuntimeVisibleAnnotations 、RuntimeInvisibleAnnotations 以及 BootstrapMethods 属性。

对于支持 class 文件主版本号(major\_version)为 49.0 或更高的 JVM，必须正确识别并读取 Signature、RuntimeVisibleAnnotations 和 RuntimeInvisibleAnnotations；

对于支持 class 文件主版本号(major\_version)为 51.0 或更高的 JVM，必须正确识别并读取 BootstrapMethods。

Java 7 规范要求任一 JVM 可以自动忽略 class 文件中它不可识别的属性项。