链滴

Java class 文件结构

作者：helly
原文链接：https://ld246.com/article/1510310081887
来源网站：链滴
许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

一、class 文件数据项的类型
u1
u2
u4
u8
...
二、class 文件的数据项
<table>
<thead>
<tr>
<th>类型</th>
<th>名称</th>
<th>数量</th>
</tr>
</thead>
<tbody>
<tr>
<td>u4</td>
<td>magic</td>
<td>1</td>
</tr>
<tr>
<td>u2</td>
<td>minor_version</td>
<td>1</td>
</tr>
<tr>
<td>u2</td>
<td>major_version</td>
<td>1</td>
</tr>
<tr>
<td>u2</td>
<td>constant_pool_count</td>
<td>1</td>
</tr>
<tr>
<td>cp_info</td>
<td>constant_pool</td>
<td>constant_pool_count - 1</td>
</tr>
<tr>
<td>u2</td>
<td>access_flags</td>
<td>1</td>
</tr>
<tr>
<td>u2</td>
<td>this_class</td>
<td>1</td>
</tr>
<tr>
<td>u2</td>
<td>super_class</td>

<td>1</td>
</tr>
<tr>
<td>u2</td>
<td>interfaces_count</td>
<td>1</td>
</tr>
<tr>
<td>u2</td>
<td>interfaces</td>
<td>interfaces_count</td>
</tr>
<tr>
<td>u2</td>
<td>fields_count</td>
<td>1</td>
</tr>
<tr>
<td>field_info</td>
<td>fields</td>
<td>fields_count</td>
</tr>
<tr>
<td>u2</td>
<td>methods_count</td>
<td>1</td>
</tr>
<tr>
<td>method_info</td>
<td>methods</td>
<td>methods_count</td>
</tr>
<tr>
<td>u2</td>
<td>attribute_count</td>
<td>1</td>
</tr>
<tr>
<td>attribute_info</td>
<td>attributes</td>
<td>attributes_count</td>
</tr>
</tbody>
</table>
三、各数据项
1、魔数（magic）
4 个字节，固定为 0XCAFEBABE。
2、次版本号（minor_version）
2 个字节。
3、主版本号（major_version）
2 个字节。
4、常量池大小（constant_pool_count）
2 个字节。
5、常量池（constant_pool）

constant_pool_count-1 个 cp_info。
索引为 0 的常量池项空白，引用索引为 0 的常量池项则表示不引用任何常量池项。
常量池数据项类型：</p>
<table>
<thead>
<tr>
<th>常量池中数据项类型</th>
<th>类型标志</th>
<th>类型描述</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONSTANT_Utf8</td>
<td>1</td>
<td>UTF-8 编码的 Unicode 字符串</td>
</tr>
<tr>
<td>CONSTANT_Integer</td>
<td>3</td>
<td>int 类型字面值</td>
</tr>
<tr>
<td>CONSTANT_Float</td>
<td>4</td>
<td>float 类型字面值</td>
</tr>
<tr>
<td>CONSTANT_Long</td>
<td>5</td>
<td>long 类型字面值</td>
</tr>
<tr>
<td>CONSTANT_Double</td>
<td>6</td>
<td>double 类型字面值</td>
</tr>
<tr>
<td>CONSTANT_Class</td>
<td>7</td>
<td>对一个类或接口的符号引用</td>
</tr>
<tr>
<td>CONSTANT_String</td>
<td>8</td>
<td>String 类型字面值</td>
</tr>
<tr>
<td>CONSTANT_Fieldref</td>
<td>9</td>
<td>对一个字段的符号引用</td>
</tr>
<tr>
<td>CONSTANT_Methodref</td>

<td>10</td>
<td>对一个类中声明的方法的符号引用</td>
</tr>
<tr>
<td>CONSTANT_InterfaceMethodref</td>
<td>11</td>
<td>对一个接口中声明的方法的符号引用</td>
</tr>
<tr>
<td>CONSTANT_NameAndType</td>
<td>12</td>
<td>对一个字段或方法的部分符号引用</td>
</tr>
</tbody>
</table>
class 文件的特殊字符串：
i. 类的全限定名：java/lang/Object
ii. 描述符：
<table>
<thead>
<tr>
<th>基本数据类型和 void 类型</th>
<th>类型的对应字符</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td>B</td>
</tr>
<tr>
<td>char</td>
<td>C</td>
</tr>
<tr>
<td>double</td>
<td>D</td>
</tr>
<tr>
<td>float</td>
<td>F</td>
</tr>
<tr>
<td>int</td>
<td>I</td>
</tr>
<tr>
<td>long</td>
<td>J</td>
</tr>
<tr>
<td>short</td>
<td>S</td>
</tr>

<tr>
<td>boolean</td>
<td>Z</td>
</tr>
<tr>
<td>void</td>
<td>V</td>
</tr>
</tbody>
</table>
引用类型：“L” + 类型的全限定名 + “;”
数组类型：若干个“[” + 数组中元素类型的对应字符串
方法：(参数 1 类型 参数 2 类型 参数 3 类型 ...)返回值类型 //各个参数类型之间没有空格， 参数列和返回值类型之间也没有空格
iii. 特殊方法的方法名
构造方法：<init>
静态代码块：<clinit>

CONSTANT_Utf8

<table>
<thead>
<tr>
<th>一个 CONSTANT_Utf8_info</th>
</tr>
</thead>
<tbody>
<tr>
<td>tag（1 个字节）</td>
</tr>
<tr>
<td>length（2 个字节）</td>
</tr>
<tr>
<td>bytes(length 个字节)</td>
</tr>
</tbody>
</table>
CONSTANT_Utf8 类型数据项可表示：程序中的字符串常量， 类型的全限定名， 方法和字段的称， 方法和字段的描述符， 属性相关字符串。

CONSTANT_NameAndType

<table>
<thead>
<tr>
<th>一个 CONSTANT_NameAndType_info</th>
</tr>
</thead>
<tbody>
<tr>
<td>tag（1 个字节）</td>
</tr>
<tr>

<td>name_index（名称）</td>
</tr>
<tr>
<td>descriptor_index（描述符）</td>
</tr>
</tbody>
</table>
一个 CONSTANT_NameAndType_info 就表示了一个被调用的方法（如父类的构造函数）或一被使用的字段。
仅在类中定义的字段或方法没有相应的 CONSTANT_NameAndType_info。

CONSTANT_Integer

<table>
<thead>
<tr>
<th>一个 CONSTANT_Integer_info</th>
</tr>
</thead>
<tbody>
<tr>
<td>tag（1 个字节）</td>
</tr>
<tr>
<td>bytes（4 个字节）</td>
</tr>
</tbody>
</table>

CONSTANT_Float

<table>
<thead>
<tr>
<th>一个 CONSTANT_Float_info</th>
</tr>
</thead>
<tbody>
<tr>
<td>tag（1 个字节）</td>
</tr>
<tr>
<td>bytes（4 个字节）</td>
</tr>
</tbody>
</table>

CONSTANT_Long

<table>
<thead>
<tr>
<th>一个 CONSTANT_Long_info</th>
</tr>

</thead>
<tbody>
<tr>
<td>tag（1 个字节）</td>
</tr>
<tr>
<td>bytes（8 个字节）</td>
</tr>
</tbody>
</table>

CONSTANT_Double

<table>
<thead>
<tr>
<th>一个 CONSTANT_Double_info</th>
</tr>
</thead>
<tbody>
<tr>
<td>tag（1 个字节）</td>
</tr>
<tr>
<td>bytes（8 个字节）</td>
</tr>
</tbody>
</table>

CONSTANT_String

<table>
<thead>
<tr>
<th>一个 CONSTANT_String_info</th>
</tr>
</thead>
<tbody>
<tr>
<td>tag（1 个字节）</td>
</tr>
<tr>
<td>string_index（2 个字节）</td>
</tr>
</tbody>
</table>

CONSTANT_Class
一个 CONSTANT_Class_info 是对类或者接口的符号引用（包括数组）。

<table>
<thead>
<tr>
<th>一个 CONSTANT_Class_info</th>

</tr>
</thead>
<tbody>
<tr>
<td>tag（1 个字节）</td>
</tr>
<tr>
<td>name_index（类的全限定名，2 个字节）</td>
</tr>
</tbody>
</table>

CONSTANT_Fieldref
该数据项表示对一个字段的符号引用， 可以是对本类中的字段的符号引用， 也可以是对其他类中的
段的符号引用， 可以是对成员变量字段的符号引用， 也可以是对静态变量的符号引用。
字段必须被使用才会有相应的符号引用，只在类中定义是没有符号引用的。

<table>
<thead>
<tr>
<th>一个 CONSTANT_Fieldref_info</th>
</tr>
</thead>
<tbody>
<tr>
<td>tag（1 个字节）</td>
</tr>
<tr>
<td>class_index（2 个字节）</td>
</tr>
<tr>
<td>name_and_type_index（2 个字节）</td>
</tr>
</tbody>
</table>

CONSTANT_Methodref
对一个类中方法的符号引用， 可以是对本类中的方法的符号引用， 也可以是对其他类中的方法的符
引用， 可以是对成员方法的符号引用， 也可以是对静态方法的符号引用，但是不会是对接口中的方
的符号引用。
方法必须被调用才会有相应的符号引用，只在类中定义是没有符号引用的。

<table>
<thead>
<tr>
<th>一个 CONSTANT_Methodref_info</th>
</tr>
</thead>
<tbody>
<tr>
<td>tag（1 个字节）</td>
</tr>
<tr>
<td>class_index（2 个字节）</td>

</tr>
<tr>
<td>name_and_type_index（2 个字节）</td>
</tr>
</tbody>
</table>

CONSTANT_InterfaceMethodref
表示对一个接口方法的符号引用。

<table>
<thead>
<tr>
<th>一个 CONSTANT_InterfaceMethodref_info</th>
</tr>
</thead>
<tbody>
<tr>
<td>tag（1 个字节）</td>
</tr>
<tr>
<td>class_index（2 个字节）</td>
</tr>
<tr>
<td>name_and_type_index（2 个字节）</td>
</tr>
</tbody>
</table>
6、当前类（或者接口）的访问修饰符（access_flags）
2 个字节
<table>
<thead>
<tr>
<th>志名</th>
<th>标志值</th>
<th>标志含义</th>
<th>针对的对像</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC_PUBLIC</td>
<td>0x0001</td>
<td>public 类型</td>
<td>所有类型</td>
</tr>
<tr>
<td>ACC_FINAL</td>
<td>0x0010</td>
<td>final 类型</td>
<td>类</td>
</tr>
<tr>
<td>ACC_SUPER</td>

<td>0x0020</td>
<td>使用新的 invokespecial 语义</td>
<td>类和接口</td>
</tr>
<tr>
<td>ACC_INTERFACE</td>
<td>0x0200</td>
<td>接口类型</td>
<td>接口</td>
</tr>
<tr>
<td>ACC_ABSTRACT</td>
<td>0x0400</td>
<td>抽象类型</td>
<td>类和接口</td>
</tr>
<tr>
<td>ACC_SYNTHETIC</td>
<td>0x1000</td>
<td>该类不由用户代码生成</td>
<td>所有类型</td>
</tr>
<tr>
<td>ACC_ANNOTATION</td>
<td>0x2000</td>
<td>注解类型</td>
<td>注解</td>
</tr>
<tr>
<td>ACC_ENUM</td>
<td>0x4000</td>
<td>枚举类型</td>
<td>枚举</td>
</tr>
</tbody>
</table>
7、对当前类的描述（this_class）（类的全限定名）
2 个字节（对一个 CONSTANT_Class_info 数据项的索引）。
8、对当前类的超类的描述（super_class）
2 个字节（对一个 CONSTANT_Class_info 数据项的索引）。
9、当前类所实现的接口的数量（interfaces_count）
注意， 只有当前类直接实现的接口才会被统计， 如果当前类继承了另一个类， 而另一个类又实现了个接口， 那么这个接口不会统计在当前类的 interfaces_count 中。
10、当前类所实现的接口（interfaces）
interfaces_count 个 u2。
每个数据项指向常量池中的一个 CONSTANT_Class_info。
11、当前的类中定义的字段的个数（fields_count）
2 个字节。
注意， 这里包括静态字段， 但不包括从父类继承的字段。 如果当前 class 文件是由一个接口生成的，那么这里的 fields_count 描述的是接口中定义的字段， 我们知道， 接口中定义的字段默认都是静态。此外要说明的是， 编译器可能会自动生成字段， 也就是说， class 文件中的字段的数量可能多于文件中定义的字段的数量。 举例来说， 编译器会为内部类增加一个字段， 这个字段是指向外围类的象的引用。
12、当前的类中定义的字段（fields）

field_count 个 field_info。
field 结构：</p>
<table>
<thead>
<tr>
<th>一个 field</th>
</tr>
</thead>
<tbody>
<tr>
<td>access_flags</td>
</tr>
<tr>
<td>name_index</td>
</tr>
<tr>
<td>descriptor_index</td>
</tr>
<tr>
<td>attributes_count</td>
</tr>
<tr>
<td>attributes</td>
</tr>
</tbody>
</table>

标志位（access_flags）
2 个字节。

<table>
<thead>
<tr>
<th>标志位名称</th>
<th>值</th>
<th>含义</th>
<th>设定者</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC_PUBLIC</td>
<td>0x0001</td>
<td>字段被设为 public</td>
<td>类和接口</td>
</tr>
<tr>
<td>ACC_PRIVATE</td>
<td>0x0002</td>
<td>字段被设为 private</td>
<td>类</td>
</tr>
<tr>
<td>ACC_PROTECTED</td>

<td>0x0004</td>
<td>字段被设为 protected</td>
<td>类</td>
</tr>
<tr>
<td>ACC_STATIC</td>
<td>0x0008</td>
<td>字段被设为 static</td>
<td>类和接口</td>
</tr>
<tr>
<td>ACC_FINAL</td>
<td>0x0010</td>
<td>字段被设为 final</td>
<td>类和接口</td>
</tr>
<tr>
<td>ACC_VOLATILE</td>
<td>0x0040</td>
<td>字段被设为 volatile</td>
<td>类</td>
</tr>
<tr>
<td>ACC_TRANSIENT</td>
<td>0x0080</td>
<td>字段被设为 transient</td>
<td>类</td>
</tr>
</tbody>
</table>

字段名（name_index）
2 个字节。
指向常量池中的一个 CONSTANT_Utf8_info 数据项。

描述符（descriptor_index）
2 个字节。
指向常量池中的一个 CONSTANT_Utf8_info 数据项。

属性数量（attributes_count）
2 个字节。

属性（attributes）
可以出现在 filed_info 中的属性有三种， 分别是 ConstantValue， Deprecated， 和 Synthetic。

13、当前的类中定义的方法的个数（methods_count）
<table>
<thead>

<tr>
<th>标志位名称</th>
<th>标志值</th>
<th>设定含义</th>
<th>设定者</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC_PUBLIC</td>
<td>0x0001</td>
<td>方法设为 public</td>
<td>类和接口</td>
</tr>
<tr>
<td>ACC_PRIVATE</td>
<td>0x0002</td>
<td>方法设为 private</td>
<td>类</td>
</tr>
<tr>
<td>ACC_PROTECTED</td>
<td>0x0004</td>
<td>方法设为 protected</td>
<td>类</td>
</tr>
<tr>
<td>ACC_STATIC</td>
<td>0x0008</td>
<td>方法设为 static</td>
<td>类</td>
</tr>
<tr>
<td>ACC_FINAL</td>
<td>0x0010</td>
<td>方法设为 final</td>
<td>类</td>
</tr>
<tr>
<td>ACC_SYNCHRONIZED</td>
<td>0x0020</td>
<td>方法设为 sychronized</td>
<td>类</td>
</tr>
<tr>
<td>ACC_NATIVE</td>
<td>0x0100</td>
<td>方法设为 native</td>
<td>类</td>
</tr>
<tr>
<td>ACC_ABSTRACT</td>
<td>0x0400</td>
<td>方法设为 abstract</td>

<td>类和接口</td>
</tr>
<tr>
<td>ACC_STRICT</td>
<td>0x0800</td>
<td>方法设为 strictFP</td>
<td>类和接口的方法</td>
</tr>
</tbody>
</table>
可以出现在 method_info 中的属性分别是 Code， Deprecated /strong>， Exceptions 和 Synthetic。
14、attribute_count
15、attribute
SourceFile // 内部类的，描述其外部类
InnerClasses // 外部类的，描述其内部类
Synchetic // 表示不是由用户代码生成的，是由编译器自动添加的，可以修饰方法和字段，例如，如一个类不定义构造方法， 那么编译器会自动添加一个无参数的构造方法， 如果定义了静态字段或静代码块， 还会根据具体情况， 增加静态初始化方法
ConstantValue // 修饰字段
Deprecated // 修饰字段和方法
符号引用：
最基本的 UTF-8 编码的 Unicode 字符串：
CONSTANT_Utf8
字面量：
CONSTATN_Integer
CONSTATN_Long
CONSTATN_Float
CONSTANT_Double
CONSTANT_String
符号引用：
CONSTANT_Class // 类或接口的符号引用
CONSTATN_NameAndType // 一个字段或方法的部分符号引用
// 以下三种符号引用由上面两种符号引用组成
CONSTATN_Fieldref // 字段的符号引用
CONSTANT_Methodref // 类中声明的方法的符号引用
CONSTANT_InterfaceMethodref // 接口中声明的方法的符号引用
#1、#2 等表示在常量池中的下标
#0 表示不引用任何常量池的项，比如
直接引用：
就是偏移量，根据这些偏移量可以找到相应的类，并在类的内存区域中找到对应的字段或方法。
class 文件其实就存了两样东西：字面量和符号引用

参考：
《深入理解 Java 虚拟机》
《Java 虚拟机规范（JavaSE8）》
http://blog.csdn.ne/brave2211/article/category/1593711/1