**孙泽嵩整理**

**1 class文件基本组成结构**

NO1. 魔数(magic)

      所有的由Java编译器编译而成的class文件的前4个字节都是“0xCAFEBABE”    
      它的作用在于：当JVM在尝试加载某个文件到内存中来的时候，会首先判断此class文件有没有JVM认为可以接受的“签名”，即JVM会首先读取文件的前4个字节，判断该4个字节是否是“0xCAFEBABE”，如果是，则JVM会认为可以将此文件当作class文件来加载并使用。

NO2.版本号(minor\_version,major\_version)

NO3.常量池计数器(constant\_pool\_count)

NO4.常量池数据区(constant\_pool[contstant\_pool\_count-1])

NO6.访问标志(access\_flags)

NO7.类索引(this\_class)

NO8.父类索引(super\_class)

NO9.接口计数器(interfaces\_count)

NO10.接口信息数据区(interfaces[interfaces\_count])

NO11.字段计数器(fields\_count)

NO12.字段信息数据区(fields[fields\_count])

NO13.方法计数器(methods\_count)

NO14.方法信息数据区(methods[methods\_count])

NO15.属性计数器(attributes\_count)

NO16.属性信息数据区(attributes[attributes\_count])

根据上述的叙述，我们可以将class的文件组织结构概括成以下面这个结构体：

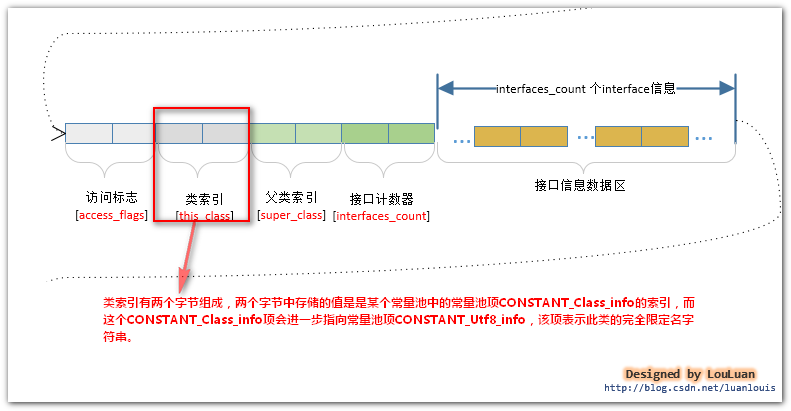
.

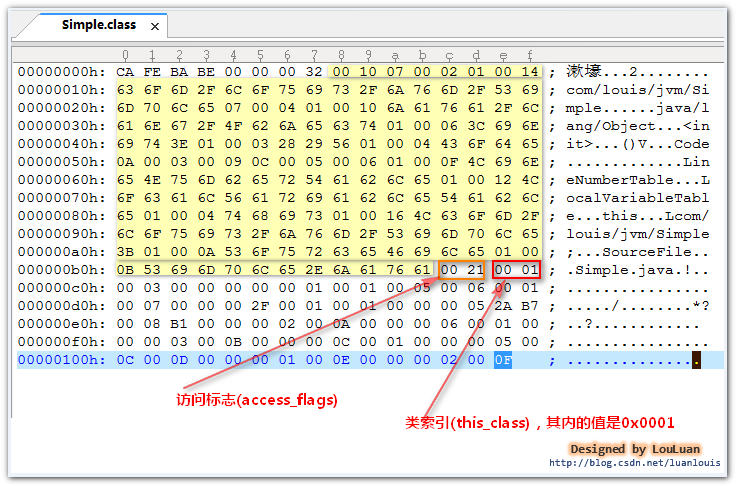
**2类索引(this\_class)**

我们知道一般情况下一个Java类源文件经过JVM编译会生成一个class文件，也有可能一个Java类源文件中定义了其他类或者内部类，这样编译出来的class文件就不止一个，但每一个class文件表示某一个类，至于这个class表示哪一个类，便可以通过 类索引 这个数据项来确定。JVM通过类的完全限定名确定是某一个类。

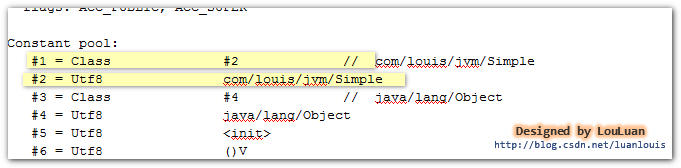
类索引的作用，就是为了指出class文件所描述的这个类叫什么名字。

类索引紧接着访问标志的后面，占有两个字节，在这两个字节中存储的值是一个指向常量池的一个索引，该索引指向的是CONSTANT

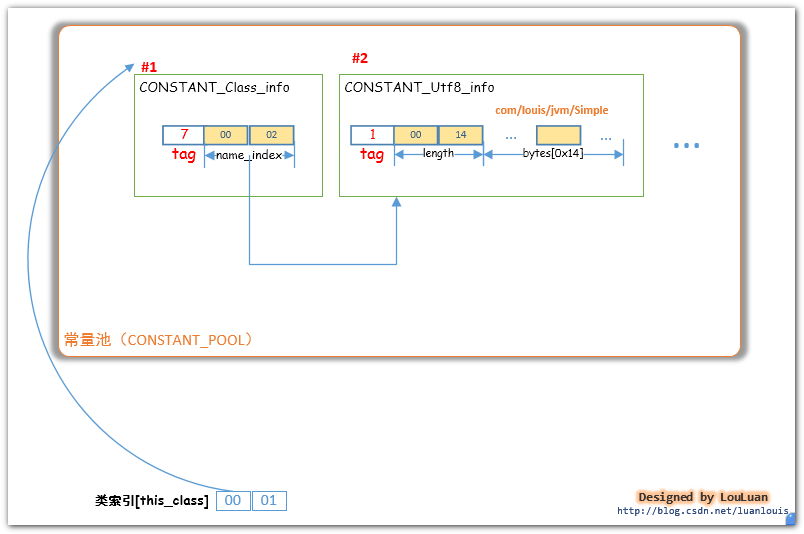
\_Class\_info常量池项，以上面定义的Simple.class 为例，如下图所示，查看他的类索引在什么位置和取什么值。



由上可知，它的类索引值为0x0001,那么，它指向了常量池中的第一个常量池项，那我们再看一下常量池中的信息。使用javap -v Simple,常量池中有以下信息：



可以看到常量池中的第一项是CONSTANT\_Class\_info项，它表示一个"com/louis/jvm/Simple"的类名。即类索引是告诉我们这个class文件所表示的是哪一个类。

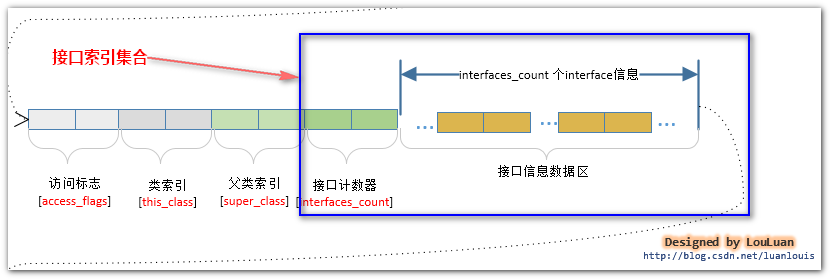


**3. 父类索引(super\_class)**

Java支持单继承模式，除了java.lang.Object 类除外，每一个类都会有且只有一个父类。class文件中紧接着类索引(this\_class)之后的两个字节区域表示父类索引，跟类索引一样，父类索引这两个字节中的值指向了常量池中的某个常量池项CONSTANT\_Class\_info，表示该class表示的类是继承自哪一个类。

**4. 接口索引集合(interfaces)**

一个类可以不实现任何接口，也可以实现很多个接口，为了表示当前类实现的接口信息，class文件使用了如下结构体描述某个类的接口实现信息:



由于类实现的接口数目不确定，所以接口索引集合的描述的前部分叫做接口计数器（interfaces\_count），接口计数器占用两个字节，其中的值表示着这个类实现了多少个接口，紧跟着接口计数器的部分就是接口索引部分了，每一个接口索引占有两个字节，接口计数器的值代表着后面跟着的接口索引的个数。接口索引和类索引和父类索引一样，其内的值存储的是指向了常量池中的常量池项的索引，表示着这个接口的完全限定名。

举例：

定义一个Worker接口，然后类Programmer实现这个Worker接口，然后我们观察Programmer的接口索引集合是怎样表示的。

**[java]** [view plain](http://blog.csdn.net/luanlouis/article/details/41039269) [copy](http://blog.csdn.net/luanlouis/article/details/41039269) [print?](http://blog.csdn.net/luanlouis/article/details/41039269)

1. /\*\*
2. \* Worker 接口类
3. \* @author luan louis
4. \*/
5. **public** **interface** Worker{
7. **public** **void** work();
9. }

**[java]** [view plain](http://blog.csdn.net/luanlouis/article/details/41039269) [copy](http://blog.csdn.net/luanlouis/article/details/41039269) [print?](http://blog.csdn.net/luanlouis/article/details/41039269)

1. **package** com.louis.jvm;
3. **public** **class** Programmer **implements** Worker {
5. @Override
6. **public** **void** work() {
7. System.out.println("I'm Programmer,Just coding....");
8. }
9. }

