第一部分

CA FE BA BE 前四个字节 魔数

第二部分：次版本号

00 00 (0)

第三部分：00 34 (54)--->1.8 51->1.7

第四部分:常量池的个数 2个字节

00 19 (25-1)=24 第0个常量被我们的jvm给占用了 表示的是什么都不引用

常量池分类 字面量以及符号引用 第一个字节是我们的的tag

java/lang/Object."<init>":()V 没有参数 返回值为null

第一个常量:

0A 00 04 00 15

methodref_info类型 0A

class_index 00 04(4)#4

name_And_Type_index 00 15(21)#21

第二个常量:

09 00 03 00 16

field_info类型 09

class_index 00 03(3)#3

name_And_Type_index 00 16(22)#22

第三个常量:

07 00 17

class_info类型 07

class_index 00 17(23)#23

第四个常量:

07 00 17

class_info类型 07

class_index 00 18(24)#24

第五个常量:

01 00 08 75 73 65 72 4E 61 6D 65

utf-8类型 01

length 00 08(8bit)

内容 75 73 65 72 4E 61 6D 65 ---------->username

第六个常量:

---------->Ljava/lang/String

第七个常量:

01 00 06 3C 69 6E 69 74 3E

utf-8类型 01

length 00 06（6bit）

内容 3C 69 6E 69 74 3E --------><init>(表示构造方法)

第八个常量:

01 00 03 28 29 56

utf-8类型 01

length 00 03（3bit）

内容 28 29 56 ---------->()V 无入参 无返回值

第九个常量: ---------->code

第十个常量: ----------> LineNumberTable

第十一个常量: ----------> LocalVariableTable

第十二个常量:

01 00 04 74 68 69 73

utf-8类型 01

length 00 04

内容 74 68 69 73 ----->this

第十三个常量:

NameAndType类型 0C

access_flag类的权限描述符(2个字节)0x0021 acc_public acc_super

This class Name:二个字节(类名索引值)00 03 com/tuling/smlz/.../***.class

super class Name:二个字节(类名索引值)00 04 java/lang/object

实现接口的个数: 二个字节: 00 00 ------->max:65535

字段的个数 00 01(1)

field_info(1):

access_flag(访问修饰符): 00 02 access_private

name_index(字段名称索引): 00 05(5)#5----->userName

descciptor(字段描述索引): 00 06(6)#6----->LJava/lang/String;

attribute_count(属性表个数):

操作数栈命令:

aload_0:将第零个参数入栈

invokespecial #1 :调用父类的构造方法 #1

return: 返回

new对象

```
0: new           #3                  // class java/util/ArrayList
3: dup
4: invokespecial #4                  // Method java/util/ArrayList."<init>":
()V
7: astore_1
8: return
```

可以看到，new字节码指令的作用是创建指定类型的对象实例、对其进行默认初始化，并且将指向该实例的一个引用压入操作数栈顶；然后因为invokespecial会消耗掉操作数栈顶的引用作为传给构造器的“this”参数，所以如果我们希望在invokespecial调用后在操作数栈顶还维持有一个指向新建对象的引用，就得在invokespecial之前先“复制”一份引用——这就是这个dup的来源。