



Java Programming

2-2

class File



Objectives

This lesson covers the following topics:

- Understand the class file structure
- Identify the access field
- Identify the method structure and bytecode
 - Method Info: Code_attribute
 - Code Attribute: LineNumberTable_attribute
 - Class Attribute: SourceFile_attribute

The class File

- Contains one Java type, either a class or an interface.
- Described using C like structures
 - Uses types u1, u2, and u4 to represent an unsigned one, two, or four byte quantity
 - No alignment or padding
 - Can be read using readUnsignedByte, readUnsignedShort, and readInt
 - Uses pseudo-array notation, even for varying size records
 - A stream of 8-bit bytes

The class File

- Java class File precisely defines the class file format, which ensures that the Java class file can be loaded and understood by any Java Virtual Machine.
- In the Java class file, the size of a variable-length item precedes the actual data for the item which allows the class file to be parsed from beginning to end with no padding.

The class File Structure

```
ClassFile {  
    u4            magic;  
    u2            minor_version;  
    u2            major_version;  
    u2            constant_pool_count;  
    cp_info       constant_pool[constant_pool_count-1];  
    u2            access_flags;  
    u2            this_class;  
    u2            super_class;  
    u2            interfaces_count;  
    u2            interfaces[interfaces_count];  
    u2            fields_count;  
    field_info     fields[fields_count];  
    u2            methods_count;  
    method_info    methods[methods_count];  
    u2            attributes_count;  
    attribute_info attributes[attributes_count];  
}
```

The java class definition contains everything a Java virtual machine needs to know about one Java class or interface.

The class File Structure

```
ClassFile {  
    u4      magic;  
    u2      minor_version;  
    u2      major_version;  
    u2      constant_pool_count;  
    cp_info  constant_pool[constant_pool_count-1];  
    u2      access_flags;  
    u2      this_class;  
    u2      super_class;  
    u2      interfaces_count;  
    u2      interfaces[interfaces_count];  
    u2      fields_count;  
    field_info fields[fields_count];  
    u2      methods_count;  
    method_info methods[methods_count];  
    u2      attributes_count;  
    attribute_info attributes[attributes_count];  
}
```

A class file consists of a Stream of 8-bit bytes.

16-bit and 32-bit quantities Are constructed by reading in two and four consecutive 8-bit bytes.

Multibyte data items are always stored in big-endian order, where the high bytes come first

The class File Structure

```
public class SampleClass {  
    public int test()  
    {  
        int x=99999;  
        int y=1;  
        int z = x + y;  
        return z;  
    }  
}
```

This is a Sample Java Source code file we will use to show the corresponding Java class File.

The SampleClass class defines only one instance method test() and 3 local Variables (x,y,z).

The class File Structure

- The sample of java bytecode for SampleClass

Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
00000000	CA	FE	BA	BE	00	00	00	34	00	10	0A	00	04	00	0D	03	0p%...4.....
00000010	00	01	86	9F	07	00	0E	07	00	0F	01	00	06	3C	69	6E	...tY.....<in
00000020	69	74	3E	01	00	03	28	29	56	01	00	04	43	6F	64	65	it>...()V...Code
00000030	01	00	0F	4C	69	6E	65	4E	75	6D	62	65	72	54	61	62	...LineNumberTab
00000040	6C	65	01	00	04	74	65	73	74	01	00	03	28	29	49	01	le...test...()I.
00000050	00	0A	53	6F	75	72	63	65	46	69	6C	65	01	00	10	53	..SourceFile...S
00000060	61	6D	70	6C	65	43	6C	61	73	73	2E	6A	61	76	61	0C	ampleClass.java.
00000070	00	05	00	06	01	00	0B	53	61	6D	70	6C	65	43	6C	61SampleCla
00000080	73	73	01	00	10	6A	61	76	61	2F	6C	61	6E	67	2F	4F	ss...java/lang/O
00000090	62	6A	65	63	74	00	21	00	03	00	04	00	00	00	00	00	bject.!.....
000000A0	02	00	01	00	05	00	06	00	01	00	07	00	00	00	1D	00
000000B0	01	00	01	00	00	00	05	2A	B7	00	01	B1	00	00	00	01*...±....
000000C0	00	08	00	00	00	06	00	01	00	00	00	01	00	01	00	09
000000D0	00	0A	00	01	00	07	00	00	00	2F	00	02	00	04	00	00/.....
000000E0	00	0B	12	02	3C	04	3D	1B	1C	60	3E	1D	AC	00	00	00<.=...`>..
000000F0	01	00	08	00	00	00	12	00	04	00	00	00	04	00	03	00
00000100	05	00	05	00	06	00	09	00	07	00	01	00	0B	00	00	00
00000110	02	00	0C														...

- This is the Binary File format produced from the Java compiler javac.

The class File Structure

1. Magic number (4 bytes)

Class files are identified by the following 4 byte header :
CAFEBABE

2. Version of class file format (4 bytes)

Minor version number of the class file format being used (2 bytes), major version number of the class file format being used(2 bytes)

The class File Structure

2. Version of class file format (4 bytes) (cont)

javase 8.0 = 52(0x 34 hex)

J2SE 6.0 = 50 (0x32 hex)

J2SE 5.0 = 49 (0x31 hex)

JDK 1.4 = 48 (0x30 hex)

JDK 1.3 = 47 (0x2F hex)

JDK 1.2 = 46 (0x2E hex)

JDK 1.1 = 45 (0x2D hex)

The class File Structure

- If a file does not start with 0XCAFEBABE, it definitely is not a valid Java class file.

```
Offset(h) 00 01 02 03
00000000  CA FE BA BE
```

- The second four bytes of a class file contain the minor and major version numbers.
- If classes have a version number that is out of the range of major or minor version then the JVM will reject to load them.

The class File Structure - Constant pool

- Following the version numbers is the constant pool.
- The Constant pool contains the constants associated with the class or interface, such as literal String, class and interface name, field name and other constants that are referred to within the class.

```
u2  constant_pool_count;
```

The class File Structure - Constant pool

- All constant_pool table entries have the following general format:

```
cp_info
{
  u1 tag;
  u1 info[];
}
```

- The constant_pool_count precedes the actual constant.
- In this example the value is 00 10, in total there are 16 entries.

CA FE BA BE 00 00 00 34 00 10 0A 00 04 00 0D 03

The class File Structure - Constant pool

The javap command displays the constant pool:

minor version: 0

major version: 52

flags: ACC_PUBLIC, ACC_SUPER

Constant pool:

```
#1 = Methodref      #4.#13      // java/lang/Object."<init>":()V
#2 = Integer        99999
#3 = Class          #14          // SampleClass
#4 = Class          #15          // java/lang/Object
#5 = Utf8           <init>
#6 = Utf8           ()V
#7 = Utf8           Code
#8 = Utf8           LineNumberTable
#9 = Utf8           test
#10 = Utf8          ()I
#11 = Utf8          SourceFile
#12 = Utf8          SampleClass.java
#13 = NameAndType   #5:#6        // "<init>":()V
#14 = Utf8          SampleClass
#15 = Utf8          java/lang/Object
```

The class File Structure - Constant pool

CA FE BA BE 00 00 00 34 00 10 0A 00 04 00 0D 03

The first constant pool entry is 0A = 10 which represents the CONSTANT_Methodref

```
CONSTANT_Methodref_info {  
  u1 tag; 0A  
  u2 class_index; 00 04  
  u2 name_and_type_index; 00 0D  
}
```


The class File Structure - Constant pool

CA FE BA BE 00 00 00 34 00 10 0A 00 04 00 0D 03

- The first entry in the constant pool is 0A which indicates the following code is for CONSTANT_Methodref.
#1 = Methodref #4.#13 // java/lang/Object."<init>":()V
- In this example, the first entry is a constructor of the class ExampleClass.
- The name of the constructor is <init> with the ()V as the descriptor.

The class File Structure - Constant pool

- Some constant pool entries refer to other locations in the class, i.e the 00 04 class_index entry refers to the index which must be a CONSTANT_Class_info structure.

```
CA FE BA BE 00 00 00 34 00 10 0A 00 04 00 0D 03
00 01 86 9F 07 00 0E 07 00 0F 01 00 06 3C 69 6E
```

- The second constant pool entry is 03 which represents the CONSTANT_Integer_info.

```
CONSTANT_Integer_info {
```

```
u1 tag; 03
```

```
u4 bytes; 00 01 86 9F
```

```
}
```

#2 = Integer 99999

In this example the 00 01 86 9F is the hex value of integer 99999.

The class File Structure – access_flags

```
00000090  62 6A 65 63 74 00 21 00 03 00 04 00 00 00 00 00
```

- u2 access_flags; ACC_PUBLIC 00 21
- Declared public; may be accessed from outside its package.

The class File Structure – this_class



00000090 62 6A 65 63 74 00 21 00 03 00 04 00 00 00 00 00

u2; 00 03

- The value of the this_class item must be a valid index into the constant_pool table.

#3 = Class #14 // SampleClass

The class File Structure – super_class



00000090 62 6A 65 63 74 00 21 00 03 00 04 00 00 00 00 00

u2 super_class; 00 04

- The value of the super_class item either must be zero or must be a valid index into the constant_pool table.

#4 = Class #15 // java/lang/Object

The class File Structure - method info

```
000000A0  02 00 01 00 05 00 06 00 01 00 07 00 00 00 1D 00
```

u2 methods_count; 02

- The method_info structures represent all methods declared by this class or interface type, including instance methods, class methods, instance initialization methods, and any class or interface initialization method .

The class File Structure - method info

```
000000A0 02 00 01 00 05 00 06 00 01 00 07 00 00 00 1D 00
```

- There are two method definitions in this SampleClass class file
- The first method – Constructor

```
02 00 01 00 05 00 06 00 01 00 07 00 00 00 1D 00
```

- The Second method – test()

```
00 08 00 00 00 06 00 01 00 00 00 01 00 01 00 09  
00 0A 00 01 00 07 00 00 00 2F 00 02 00 04 00 00
```

The class File Structure - method info

The first method – Constructor

02 00 01 00 05 00 06 00 01 00 07 00 00 00 10 00

```
method_info {  
  u2 access_flags; 00 01  
  u2 name_index; 00 05  
  u2 descriptor_index; 00 06  
  u2 attributes_count; 00 01  
  attribute_info attributes[attributes_count];  
}  
ACC_PUBLIC 0x0001 Declared public; may be accessed from outside its  
package.
```


The class File Structure - method info

The first method – Constructor

```
02 00 01 00 05 00 06 00 01 00 07 00 00 00 1D 00
```

name_index 00 05

The value of the name_index item must be a valid index into the constant_pool table.

#5 = Utf8 <init>

This method is the instance initialization method which has special name <init>.

The class File Structure - method info

- The first method – Constructor

```
02 00 01 00 05 00 06 00 01 00 07 00 00 00 1D 00
```

- The name is supplied by a compiler.
- Because the name <init> is not a valid identifier, it cannot be used directly in a program written in the Java programming language.

The class File Structure - method info

The first method – Constructor

02 00 01 00 05 00 06 00 01 00 07 00 00 00 10 00

```
method_info {  
  u2 access_flags; 00 01  
  u2 name_index; 00 05  
  u2 descriptor_index; 00 06  
  u2 attributes_count; 00 01  
  attribute_info attributes[attributes_count];  
}  
descriptor_index 00 06
```

The value of the descriptor_index item must be a valid index into the constant_pool table. #6 = Utf8 ()V

The class File Structure - method info

The first method – Constructor

02 00 01 00 05 00 06 00 01 00 07 00 00 00 1D 00

attributes_count 00 01

- The value of the attributes_count item indicates the number of additional attributes of this method.
- In this example, there is only one attribute information for this constructor.

The class File Structure - method info

The first method – Constructor Attribute

```
02 00 01 00 05 00 08 00 01 00 07 00 00 00 1D 00
01 00 01 00 00 00 05 2A B7 00 01 B1 00 00 00 01
```

```
Code_attribute { u2 attribute_name_index; 00 07 (code)
u4 attribute_length; 00 00 00 1D (29)
u2 max_stack; 00 01
u2 max_locals; 00 01
u4 code_length; 00 00 00 05
u1 code[code_length]; 2A B7 00 01 B1
u2 exception_table_length; { u2 start_pc; u2 end_pc; u2 handler_pc; u2
catch_type; } exception_table[exception_table_length]; 00
u2 attributes_count; 01
attribute_info attributes[attributes_count]; }
```

The class File Structure - method info

- The first method – Constructor Attribute

```
02 00 01 00 05 00 08 00 01 00 07 00 00 00 1D 00  
01 00 01 00 00 00 05 2A B7 00 01 B1 00 00 00 01
```

- A Code attribute contains the Java Virtual Machine instructions and auxiliary information for a method.
attribute_name_index
- The value of the attribute_name_index item must be a valid index into the constant_pool table.
- In this example, 00 07 indicates a Code constant.

#7 = Utf8

Code

The class File Structure - method info

- The first method – Constructor Attribute

```
02 00 01 00 05 00 08 00 01 00 07 00 00 00 1D 00  
01 00 01 00 00 00 05 2A B7 00 01 B1 00 00 00 01
```

u2 max_stack; 00 01

- The maximum depth of the operand stack of this method is 1.

u2 max_locals; 00 01

- The number of local variables is 1.

The class File Structure - method info

- The first method – Constructor Attribute

```
02 00 01 00 05 00 06 00 01 00 07 00 00 00 1D 00
01 00 01 00 00 00 05 2A B7 00 01 B1 00 00 00 01
```

u4 code_length; 00 00 00 05

- The number of bytes in the code array.

u1 code[5]; 2A B7 00 01 B1

0: aload_0

1: invokespecial #1

4: return

// Method java/lang/Object."<init>":()V

The class File Structure - method info

The Second method – test()

```
00 08 00 00 00 06 00 01 00 00 00 01 00 01 00 09
00 0A 00 01 00 07 00 00 00 2F 00 02 00 04 00 00
```

```
method_info {
  u2 access_flags; 00 01
  u2 name_index; 00 09
  u2 descriptor_index; 00 0A
  u2 attributes_count; 00 01
  attribute_info attributes[attributes_count];
}
```

The class File Structure - method info

- The Second method – test()

```
00 08 00 00 00 06 00 01 00 00 00 01 00 01 00 09
00 0A 00 01 00 07 00 00 00 2F 00 02 00 04 00 00
```

ACC_PUBLIC0x0001

- Declared public; may be accessed from outside its package.

name_index 00 09

- The value of the name_index item must be a valid index into the constant_pool table.

The class File Structure - method info

- The Second method – test()

```
00 08 00 00 00 06 00 01 00 00 00 01 00 01 00 09
00 0A 00 01 00 07 00 00 00 2F 00 02 00 04 00 00
```

- The entry to the constant pool is 9.
#9 = Utf8 test
- This is a normal Java instance method with the name of test().

The class File Structure - method info

- The Second method – test()

```
00 08 00 00 00 06 00 01 00 00 00 01 00 01 00 09  
00 0A 00 01 00 07 00 00 00 2F 00 02 00 04 00 00
```

descriptor_index 00 06A

- The value of the descriptor_index item must be a valid index in the constant_pool table.

#10 = Utf8 ()V

The class File Structure - method info

- The Second method – test()

```
00 08 00 00 00 06 00 01 00 00 00 01 00 01 00 09
00 0A 00 01 00 07 00 00 00 2F 00 02 00 04 00 00
```

attributes_count 00 01

- The value of the attributes_count item indicates the number of additional attributes of this method.
- In this example, there is only one attribute information for the test() method.

The class File Structure - method info

The Second method – test() method Attribute

00	0A	00	01	00	07	00	00	00	2F	00	02	00	04	00	00
00	0B	12	02	3C	04	3D	1B	1C	60	3E	1D	AC	00	00	00
01	00	08	00	00	00	12	00	04	00	00	00	04	00	03	00
05	00	05	00	08	00	09	00	07	00	01	00	0B	00	00	00

Code_attribute { u2 attribute_name_index; 00 07 (code)

u4 attribute_length; 00 00 00 2F (47 bytes)

u2 max_stack; 00 02

u2 max_locals; 00 04

u4 code_length; 00 00 00 0B

u1 code[11]; 12 02 3B 04 3C 1A 1B 60 3D 1C Ac

u2 exception_table_length; { u2 start_pc; u2 end_pc; u2 handler_pc; u2 catch_type; } exception_table[exception_table_length]; 00

u2 attributes_count; 01

attribute_info attributes[attributes_count]; }

The class File Structure - method info

- The Second method – test() method Attribute

```
00 0B 12 02 3C 04 3D 1B 1C 60 3E 1D AC 00 00 00
01 00 08 00 00 00 12 00 04 00 00 00 04 00 03 00
05 00 05 00 06 00 09 00 07 00 01 00 0B 00 00 00
```

```
LineNumberTable_attribute {
    u2 attribute_name_index;
    u4 attribute_length;
    u2 line_number_table_length;
    { u2 start_pc;
      u2 line_number;
    } line_number_table[line_number_table_length];
}
```

The class File Structure - method info

- The Second method – test() method Attribute

```
00 0B 12 02 3C 04 3D 1B 1C 60 3E 1D AC 00 00 00
01 00 08 00 00 00 12 00 04 00 00 00 04 00 03 00
05 00 05 00 06 00 09 00 07 00 01 00 0B 00 00 00
```

attribute_name_index : 00 08

#8 = Utf8 LineNumberTable

attribute_length: 00 00 00 12

The class File Structure - method info

- SampleClass class Attribute



A hex dump visualization of class attribute data. The first row contains 16 bytes: 05 00 05 00 06 00 09 00 07 00 01 00 08 00 00 00. The second row contains 4 bytes: 02 00 0C followed by a black square, then a grey rectangle. The hex values are displayed in purple on a blue background for the first row and white on a blue background for the second row.

```
SourceFile_attribute {  
    u2 attribute_name_index;  
    u4 attribute_length;  
    u2 sourcefile_index;  
}
```

The class File Structure - method info

- SampleClass class Attribute

```
05 00 05 00 06 00 09 00 07 00 01 00 0B 00 00 00  
02 00 0C
```

attribute_name_index: 00 0B

#11 = Utf8 SourceFile

attribute_length 00 00 00 02

sourcefile_index : 00 0C

The value of the sourcefile_index item must be a valid index in the constant_pool table

#12 = Utf8 SampleClass.java

Summary

In this lesson, you should have learned:

- Understand the class file structure
- Identify the access field
- Identify the method structure and bytecode
 - Method Info: Code_attribute
 - Code Attribute: LineNumberTable_attribute
 - Class Attribute: SourceFile_attribute

