

Leitor Exibidor Java

Grupo 5

Gabriel Martins de Miranda - 130111350

Marina Martins de Miranda - 110132351

Victor Fernandes Uriarte - 110021193

Guilherme Neves Souza - 130113182

Elton Araujo de Castro - 110028384

Java 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];  
}
```

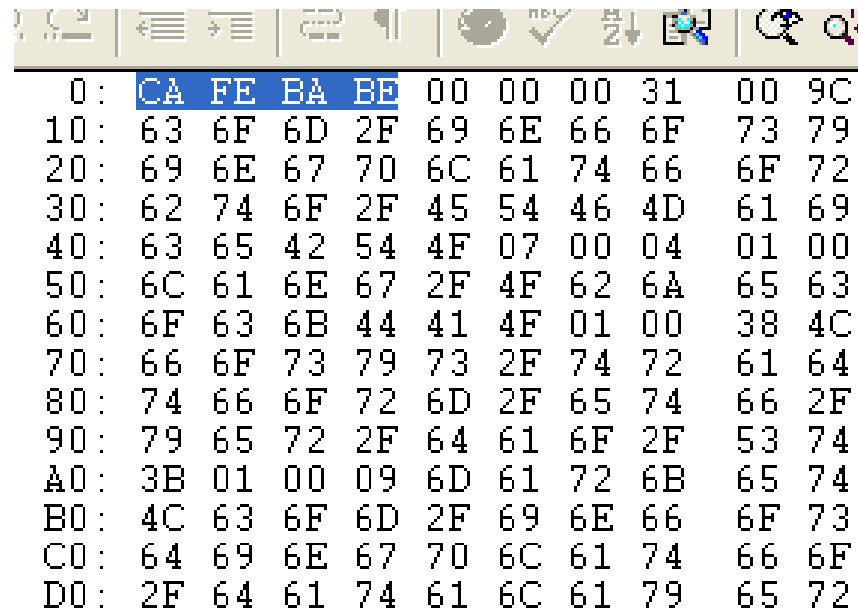
Magic	Version
Constant Pool	
Access Flags	
this Class	
super Class	
Interfaces	
Fields	
Methods	
Attributes	

Java Class File structure

- A Java class file is consist of 10 basic sections:
- Magic Number: 0xCAFEBAFE
- Version of Class File Format: the minor and major versions of the class file
- Constant Pool: Pool of constants for the class
- Access Flags: for example whether the class is abstract, static, etc.
- This Class: The name of the current class
- Super Class: The name of the super class
- Interfaces: Any interfaces in the class
- Fields: Any fields in the class
- Methods: Any methods in the class

Magic number

- **Magic number** is used to uniquely identify the format and to distinguish it from other formats.
- The first four bytes of the Class file are **0xCAFEBABE**.



0:	CA	FE	BA	BE	00	00	00	31	00	9C
10:	63	6F	6D	2F	69	6E	66	6F	73	79
20:	69	6E	67	70	6C	61	74	66	6F	72
30:	62	74	6F	2F	45	54	46	4D	61	69
40:	63	65	42	54	4F	07	00	04	01	00
50:	6C	61	6E	67	2F	4F	62	6A	65	63
60:	6F	63	6B	44	41	4F	01	00	38	4C
70:	66	6F	73	79	73	2F	74	72	61	64
80:	74	66	6F	72	6D	2F	65	74	66	2F
90:	79	65	72	2F	64	61	6F	2F	53	74
A0:	3B	01	00	09	6D	61	72	6B	65	74
B0:	4C	63	6F	6D	2F	69	6E	66	6F	73
C0:	64	69	6E	67	70	6C	61	74	66	6F
D0:	2F	64	61	74	61	6C	61	79	65	72

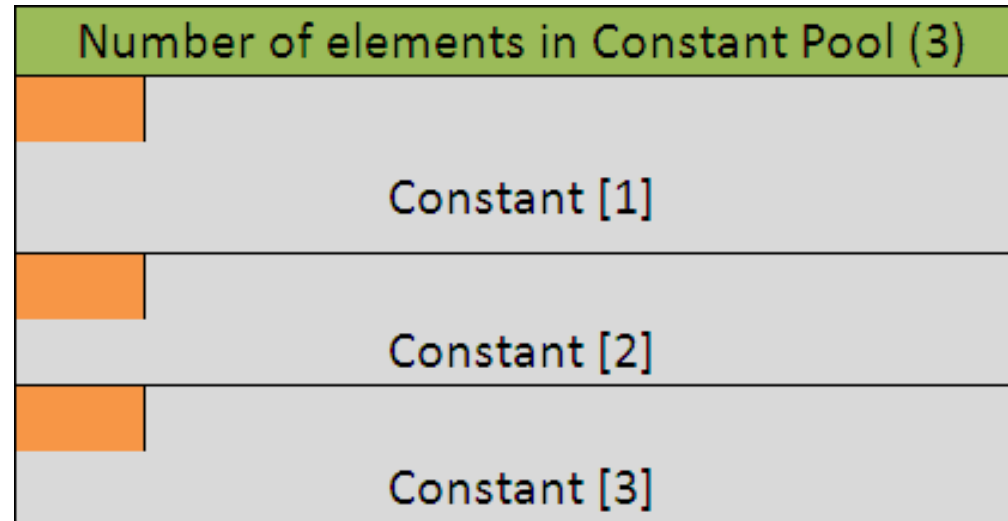
Version of Class file

- The next four byte of the class file contains major and minor version numbers. This number allows the JVM to verify and identify the class file. If the number is greater than what JVM can load, the class file will be rejected with error message: `UnsupportedClassVersionError`.

Major Version	Hex	JDK version
51	0x33	J2SE 7
50	0x32	J2SE 6.0
49	0x31	J2SE 5.0
48	0x30	JDK 1.4
47	0x2F	JDK 1.3
46	0x2E	JDK 1.2
45	0x2D	JDK 1.1

Constant Pool

- All the constants related to the Class or an Interface will get stored in the Constant Pool. The constants includes class names, variable names, interface names, method names and signature, final variable values, string literals etc.



Constant Pool

Java Virtual Machine instructions do not rely on the run-time layout of classes, interfaces, class instances, or arrays. Instead, instructions refer to symbols and other information in the constant_pool.

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

Table 4.3. Constant pool tags

Constant Type	Value
CONSTANT_Class	7
CONSTANT_Fieldref	9
CONSTANT_Methodref	10
CONSTANT_InterfaceMethodref	11
CONSTANT_String	8
CONSTANT_Integer	3
CONSTANT_Float	4
CONSTANT_Long	5
CONSTANT_Double	6
CONSTANT_NameAndType	12
CONSTANT_Utf8	1
CONSTANT_MethodHandle	15
CONSTANT_MethodType	16
CONSTANT_InvokeDynamic	18

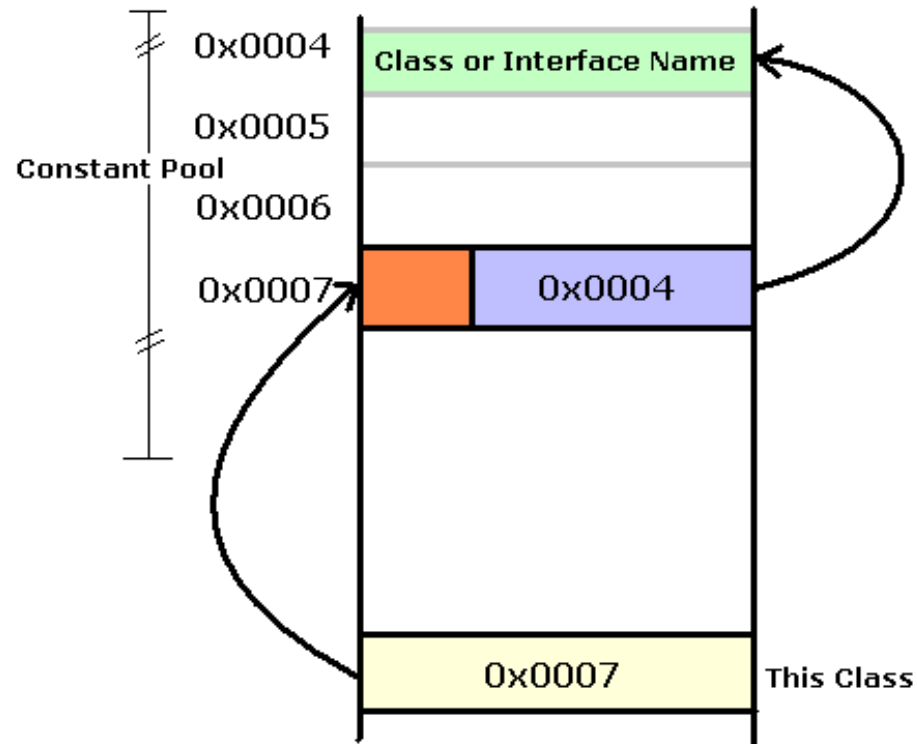
Access flags

- Access flags follows the Constant Pool. It is a two byte entry that indicates whether the file defines a class or an interface, whether it is public or abstract or final in case it is a class. Below is a list of some of the access

Flag Name	Value	Interpretation
ACC_PUBLIC	0x0001	Declared public ; may be accessed from outside its package.
ACC_FINAL	0x0010	Declared final ; no subclasses allowed.
ACC_SUPER	0x0020	Treat superclass methods specially when invoked by the invokespecial instruction.
ACC_INTERFACE	0x0200	Is an interface , not a class.
ACC_ABSTRACT	0x0400	Declared abstract ; may not be instantiated.

this Class

- Access flags follows the Constant Pool. It is a two byte entry that indicates whether the file defines a class or an interface, whether it is public or abstract or final in case it is a class. Below is a list of some of the access flags and their interpretation.



super Class

- Next 2 bytes after This Class is of Super Class. Similar to this class, value of two bytes is a pointer that points to Constant pool which has entry for super class of the class.

Interfaces

- All the interfaces that are implemented by the class (or interface) defined in the file goes in Interface section of a class file. Starting two byte of the Interface section is the count that provides information about total number of interfaces being implemented. Immediately following is an array that contains one index into the constant pool for each interface implemented by class.

Fields

- A field is an instance or a class level variable (property) of the class or interface. Fields section contains only those fields that are defined by the class or an interface of the file and not those fields which are inherited from the super class or super interface.
- First two bytes in Fields section represents count: that is the total number of fields in Fields Section. Following the count is an array of variable length structure one for each field. Each element in this array represent one field. Some information is stored in this structure where as some information like name of the fields are stored in Constant pool

Fields

```
field_info {  
    u2          access_flags;  
    u2          name_index;  
    u2          descriptor_index;  
    u2          attributes_count;  
    attribute_info attributes[attributes_count];  
}
```

Table 4.4. Field access and property flags

Flag Name	Value	Interpretation
ACC_PUBLIC	0x0001	Declared public; may be accessed from outside its package.
ACC_PRIVATE	0x0002	Declared private; usable only within the defining class.
ACC_PROTECTED	0x0004	Declared protected; may be accessed within subclasses.
ACC_STATIC	0x0008	Declared static.
ACC_FINAL	0x0010	Declared final; never directly assigned to after object construction (JLS §17.5).
ACC_VOLATILE	0x0040	Declared volatile; cannot be cached.
ACC_TRANSIENT	0x0080	Declared transient; not written or read by a persistent object manager.
ACC_SYNTHETIC	0x1000	Declared synthetic; not present in the source code.
ACC_ENUM	0x4000	Declared as an element of an enum.

Methods

- The Methods component host the methods that are explicitly defined by this class, not any other methods that may be inherited from super class.
- First two byte is the count of the number of methods in the class or interface. The rest is again a variable length array which holds each method structure. Method structure contains several pieces of information about the method like method argument list, its return type, the number of stack words required for the method's local variables, stack words required for method's operand stack, a table for exceptions, byte code sequence etc

Methods

```
method_info {  
    u2          access_flags;  
    u2          name_index;  
    u2          descriptor_index;  
    u2          attributes_count;  
    attribute_info attributes[attributes_count];  
}
```

Table 4.5. Method access and property flags

Flag Name	Value	Interpretation
ACC_PUBLIC	0x0001	Declared public; may be accessed from outside its package.
ACC_PRIVATE	0x0002	Declared private; accessible only within the defining class.
ACC_PROTECTED	0x0004	Declared protected; may be accessed within subclasses.
ACC_STATIC	0x0008	Declared static.
ACC_FINAL	0x0010	Declared final; must not be overridden (§5.4.5).
ACC_SYNCHRONIZED	0x0020	Declared synchronized; invocation is wrapped by a monitor use.
ACC_BRIDGE	0x0040	A bridge method, generated by the compiler.
ACC_VARARGS	0x0080	Declared with variable number of arguments.
ACC_NATIVE	0x0100	Declared native; implemented in a language other than Java.
ACC_ABSTRACT	0x0400	Declared abstract; no implementation is provided.
ACC_STRICT	0x0800	Declared strictfp; floating-point mode is FP-strict.
ACC_SYNTHETIC	0x1000	Declared synthetic; not present in the source code.

Attributes

- Attribute section contains several attribute about the class file, e.g. one of the attribute is the source code attribute which reveals the name of the source file from which this class file was compiled.
- First two bytes in Attribute section is count of the number of attributes, followed by the attributes themselves. The JVMs will ignore any attributes they don't understand.
- Let me know your comments and suggestions about this tutorial.

Attributes

```
attribute_info {  
    u2 attribute_name_index;  
    u4 attribute_length;  
    u1 info[attribute_length];  
}
```

Table 4.6. Predefined class file attributes

Attribute	Section	Java SE	class file
ConstantValue	§4.7.2	1.0.2	45.3
Code	§4.7.3	1.0.2	45.3
StackMapTable	§4.7.4	6	50.0
Exceptions	§4.7.5	1.0.2	45.3
InnerClasses	§4.7.6	1.1	45.3
EnclosingMethod	§4.7.7	5.0	49.0
Synthetic	§4.7.8	1.1	45.3
Signature	§4.7.9	5.0	49.0
SourceFile	§4.7.10	1.0.2	45.3
SourceDebugExtension	§4.7.11	5.0	49.0
LineNumberTable	§4.7.12	1.0.2	45.3
LocalVariableTable	§4.7.13	1.0.2	45.3
LocalVariableTypeTable	§4.7.14	5.0	49.0
Deprecated	§4.7.15	1.1	45.3
RuntimeVisibleAnnotations	§4.7.16	5.0	49.0
RuntimeInvisibleAnnotations	§4.7.17	5.0	49.0
RuntimeVisibleParameterAnnotations	§4.7.18	5.0	49.0
RuntimeInvisibleParameterAnnotations	§4.7.19	5.0	49.0
AnnotationDefault	§4.7.20	5.0	49.0
BootstrapMethods	§4.7.21	7	51.0

Bibliografia

- <http://viralpatel.net/blogs/tutorial-java-class-file-format-revealed/>
- <https://docs.oracle.com/javase/specs/jvms/se7/html/jvms-4.html>
- <https://docs.oracle.com/javase/specs/jvms/se7/html/jvms-6.html>