Everything you always wanted to know about Java Class Versions

Margrit Hoehme (2008, java@mhoehme.de)

What does UnsupportedClassVersionError mean?

Probably you have already run into an error like this:

```
Exception in thread "main" java.lang.UnsupportedClassVersionError: MyClass
(Unsupported major.minor version 50.0)
```

This error is “thrown when the Java Virtual Machine attempts to read a class file and determines that the major and minor version numbers in the file are not supported“ (http://java.sun.com/javase/6/docs/api/index.html).

In other words: There is a mismatch between the JVM version and the class file version. This error typically occurs when an older JVM is used to run a class file compiled by a more recent compiler.

Checking the JVM version

Finding out the java runtime version is quite easy:

```
$ java -version
java version "1.3.1_01"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.3.1_01)
Java HotSpot(TM) Client VM (build 1.3.1_01, mixed mode)
```

But what is the compiler version of the class?

Getting the Class Version

Finding out the compiler version of the class is a little bit more tricky than finding out the java runtime version.

What is the class version?

When we look at the format of a class file, we find the following (Java Virtual Machine Specification, Ch. 4, The Class File Format):

magic (CAFEBABE)		**minor version**	**major version**
constant pool count	**[constant pool]**		
access flags (eg public)	**this class**	**super class**	**interfaces count**
[interfaces]			
field count	**[fields]**		
method count	**[methods]**		
attributes count	**[attributes]**		

The actual length of the fields in brackets depends on the value of the preceding `count` field.

The magic values identifies a valid class file. The minor and major versions together are the class file version.

Java Virtual Machines may support only a range of file format versions. When the class file version is outside of this range, the UnsupportedClassVersionError is thrown.

Supported ranges of class file versions of Sun JVMs are:

- JDK 1.0.2 = 45.0 – 45.3
- JDK 1.1.X = 45.0 – 45.65535
- JDK 1.2 = 45.0 – 46.0
- JDK 1.3 = 47
- JDK 1.4 = 48
- J2SE 5.0 = 49
- J2SE 6.0 = 50

How to read the class version from the class file

Now that we know the file format and where to find the class version, it should be quite easy to read the class file version from a given classfile:

```java
DataInputStream dis = new DataInputStream(new FileInputStream(path));

if (dis.readInt() != 0xCAFEBABE) {
   dis.close();
   throw new IOException
      ("Magic is not 0xCAFEBABE - not a valid class file ["+path+"]");
}

int minor = dis.readUnsignedShort();
int major = dis.readUnsignedShort();

dis.close();

String version = major + "." + minor;
```

From JDK 1.5 onwards, a `ClassParser` can be used to obtain the same result:

```
import com.sun.org.apache.bcel.internal.classfile.ClassParser;
import com.sun.org.apache.bcel.internal.classfile.JavaClass;

JavaClass c = new ClassParser(new FileInputStream(path), path).parse();
String version =  c.getMajor()+"."+c.getMinor();
```

Resources and further Reading

- The Java Virtual Machine Specification (2nd edition): http://java.sun.com/docs/books/jvms/second_edition/html/VMSpecTOC.doc.html
- http://www.javaworld.com/javaworld/jw-07-1996/jw-07-classfile.html
- http://java.sun.com/javase/6/docs/api/index.html