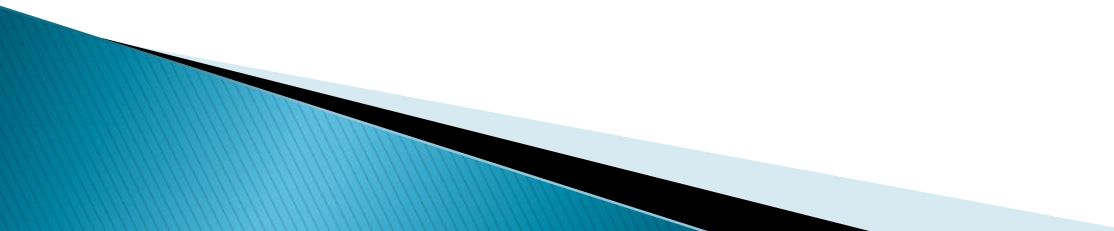


Bytecode with ASM

Michael Rasmussen
ZeroTurnaround

Today's Outline

- ▶ Terminology
 - ▶ ASM basics
 - ▶ Generating bytecode
 - Useful examples
 - Control flow templates
 - ▶ Bytecode in the wild!
- 

Terminology

- »» Lately I've been,
I've been losing sleep,
dreaming about the things
that we could be

Binary Name

- ▶ String representing a class' name
- ▶ Fully qualified class name
 - uses / as package separator
- ▶ Example:
 - java.lang.String “java/lang/String”
 - java.util.ArrayList “java/util/ArrayList”

Type Descriptor

- ▶ Internal definition of a type
- ▶ Primitive types
 - Single character
 - “J” (long), “I” (int), “S” (short), “B” (byte), “C” (char)
 - “F” (float), “D” (double)
 - “Z” (boolean), “V” (void)

Type Descriptor

▶ Object types

- Binary name enclosed by L;
 - “Ljava/lang/String;”
 - “Ljava/util/ArrayList;”

▶ Array types

- [followed by type descriptor
 - “[B” byte[]
 - “[Ljava/lang/String;” String[]
 - “[[D” double[][]
 - Multiple [indicates multiple dimensions

Method Descriptor

- ▶ Defines parameter and return types
- ▶ Consists of:
 - Enclosed in parenthesis: 0 or more Type Descriptors
 - Describing the parameters
 - 1 Type Descriptor
 - Describing the return type

Method Descriptor

▶ Example

- “(Ljava/lang/String;)I”
 - Takes a java.lang.String as arguments
 - Returns an int
- “()V”
 - Takes zero arguments
 - Returns nothing (void)
- “(DD)D”
 - Takes two arguments of type double
 - Returns a double

Special Methods

▶ Constructor

- Special name: “<init>”
- **Must** invoke constructor on superclass (or another constructor on this class)

▶ Static initializer

- Special name: “<clinit>”
- Method invoked when the class is initialized
 - Ensured to only be called once
 - Ensured to have been called before any methods on the class is called

Access / Modifiers

▶ Bit-mask containing access flags

- ACC_PUBLIC class, field, method
- ACC_PRIVATE class, field, method
- ACC_PROTECTED class, field, method
- ACC_STATIC field, method
- ACC_FINAL class, field, method, parameter
- ACC_SUPER class
- ACC_SYNCHRONIZED method
- ACC_VOLATILE field
- ACC_BRIDGE method
- ACC_VARARGS method
- ACC_TRANSIENT field
- ACC_NATIVE method
- ACC_INTERFACE class
- ACC_ABSTRACT class, method
- ACC_STRICT method
- ACC_SYNTHETIC class, field, method, parameter
- ACC_ANNOTATION class
- ACC_ENUM class, field
- ACC_MANDATED parameter

Basic ASM Classes

»» Do I look good for you tonight
Will you accuse me as I hide
Behind these layers of disguise

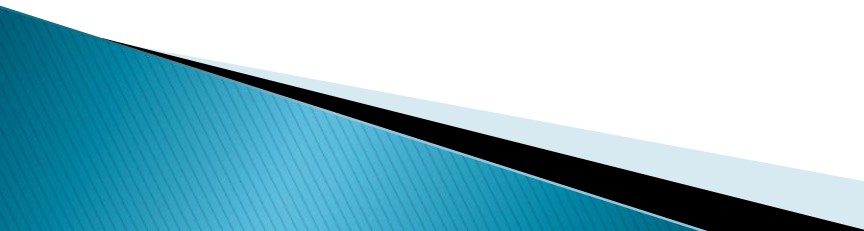
ClassVisitor

- ▶ Base Visitor to visit a class
 - Contains visitors for (among others):
 - Describing the class, superclass, interfaces
 - Visiting fields
 - Visiting methods

ClassVisitor

▶ visit(...)

◦ Visits the overall class

- int version Java version
 - int access Access for class (ACC_PUBLIC)
 - String name Name of class
 - String signature Generic signature
 - String superName Name of superclass
 - String[] interfaces Names of implemented interfaces
- 

ClassVisitor

▶ visitField(...) : FieldVisitor

◦ Visit a field on the class

- int access Access for the field (public/private)
- String name Name of the field
- String desc Type descriptor of the field
- String signature Generic signature for the field
- Object value Default value (for static primitives)

ClassVisitor

▶ visitMethod(...) : MethodVisitor

◦ Visit a method on the class

- int access Access for the method
- String name Name of the method
- String desc Method descriptor of the method
- String signature Generic signature for the method
- String[] exceptions Declared thrown exceptions

ClassWriter

- ▶ Extends ClassVisitor
- ▶ Basic class to generate a class
- ▶ Use `toByteArray()` to get actual bytecode

ClassReader

- ▶ Basic class to parse a class
- ▶ Use `accept()` with a `ClassVisitor`
 - visits on the passed `ClassVisitor` the individual parts of the class being parsed

Label

- ▶ Symbolizes branch targets in methods

- Example:

- `Label labelGoto = new Label();`
`mv.visitJumpInsn(GOTO, labelGoto);`
`mv.visitLabel(labelGoto);`

MethodVisitor

- ▶ Base Visitor to visit a method
- ▶ Has visit methods for individual bytecode opcode-groups

MethodVisitor

- ▶ visitInsn(int opcode)
 - Visits a zero operand instruction
 - xRETURN, ATHROW
 - xCONST_n
 - xALOAD, xASTORE, ARRAYLENGTH
 - xADD, xSUB, xMUL, xDIV, xREM, xNEG
 - xOR, xAND, xXOR, xSHL, xSHR, xUSHR
 - I2x, L2x, F2x, D2x
 - LCMP, FCMPx, DCMPx
 - NOP, SWAP, DUP, DUP2, POP, POP2, DUP_x
 - MONITORENTER, MONITOREXIT

MethodVisitor

- ▶ visitIntInsn(int opcode, int operand)
 - Visits an instruction with a single int operand
 - BIPUSH, SIPUSH
 - Pushed the value “operand” onto the stack
 - NEWARRAY
 - Creates a primitive array of the type specified by “operand”:
 - T_BOOLEAN, T_CHAR, T_FLOAT, T_DOUBLE, T_BYTE, T_SHORT, T_INT, T_LONG
 - Size of array is popped from the stack

MethodVisitor

- ▶ `visitVarInsn(int opcode, int var)`
 - Visits a local instruction
 - `xLOAD`
 - `xSTORE`
 - “var” indicates the local index to access

MethodVisitor

- ▶ visitTypeInsn(int opcode, String type)
 - Visits a type instruction.
 - NEW
 - ANEWARRAY
 - CHECKCAST
 - INSTANCEOF
 - “type” is the type’s binary name

MethodVisitor

- ▶ visitFieldInsn(int opcode, String owner, String name, String desc)
 - Visits a field instruction
 - GETSTATIC, PUTSTATIC
 - GETFIELD, PUTFIELD
 - “owner” is the binary-name of the class that holds the field
 - “name” is the name of the field
 - “desc” is the type-descriptor for the field

MethodVisitor

- ▶ visitMethodInsn(int opcode, String owner, String name, String desc, boolean itf)
 - Visits a method instruction
 - INVOKESTATIC
 - INVOKEVIRTUAL, INVOKESPECIAL, INVOKEINTERFACE
 - “owner” is the binary-name of the class that holds the method
 - “name” is the name of the method
 - “desc” is the method-descriptor of the method
 - “itf” indicates if it’s an interface-method

MethodVisitor

- ▶ visitJumpInsn(int opcode, Label label)
 - Visits a jump instruction.
 - GOTO, IF_ICMPx, IFx, IF_ACMPx, etc
 - “label” points to the target location to jump to if condition is met

MethodVisitor

- ▶ visitLdcInsn(Object cst)
 - Visits an LDC instruction.
 - LDC
 - If “cst” is of the type Integer, Long, Float or Double, the value of that number is used as a constant.
 - If “cst” is of type String, the content of that String is used as a constant.
 - If “cst” is of type Type, the type indicated by that Type is used as a constant (equivalent to using .class in Java source code)

MethodVisitor

- ▶ `visitInsn(int var, int increment)`
 - Visits an IINC instruction.
 - IINC
 - “var” points to the local index to modify
 - “increment” is the value the local should be incremented by (valid range: -32768 .. 32767)

MethodVisitor

- ▶ visitLabel(Label label)
 - Visits a label.
 - Adds the target location “label”, for use with, among others, jump opcodes.

Type

- ▶ ASM class for type/method descriptors
 - Contains several useful helper methods
 - `Type.getInternalName(Class c) : String`
 - Returns the binary-name for a class
 - `Type.getDescriptor(Class c) : String`
 - Returns the type-description for a class
 - `Type.getType(Class c) : Type`
 - Return a Type object for the specified class
 - `Type.getMethodDescriptor(Type returnType, Type... paramTypes) : String`
 - Returns the method-descriptor for the types specified

Generating bytecode

»» We built this city,
We built this city on rock and roll

Generating bytecode

▶ Basics for generating a class

```
◦ ClassWriter cw = new ClassWriter();

cw.visit(V1_5, ACC_PUBLIC, "className", null,
        Type.getInternalName(Object.class), null);

// Visit individual fields and methods
// cw.visitField(...)
// cw.visitMethod(...)

cw.visitEnd();

byte[] classBytes = cw.toByteArray();
// define classBytes using a ClassLoader
```


Generating bytecode

▶ Basics for generating a method

- `MethodVisitor mv = cw.visitMethod(ACC_PUBLIC | ACC_STATIC, "main", Type.getMethodDescriptor(Type.VOID_TYPE, Type.getType(String[].class)), null, null);`

```
mv.visitFieldInsn(GETSTATIC,
    Type.getInternalName(System.class),
    "out",
    Type.getDescriptor(PrintStream.class));
```

```
mv.visitLdcInsn("Hello World!");
```

```
mv.visitMethodInsn(INVOKEVIRTUAL,
    Type.getInternalName(PrintStream.class),
    "println",
    Type.getMethodDescriptor(Type.VOID_TYPE, Type.getType(String.class)),
    false);
```

```
mv.visitInsn(RETURN);
```

```
mv.visitMaxs(2, 1);
mv.visitEnd();
```

Section-subscript credits

- ▶ OneRepublic – Counting Stars
- ▶ Manic Street Preachers – Born a Girl
- ▶ Starship – We Built this City