Let's make some 0xCAFEBABE

Test-Driven Bytecode Engineering

Evgeny Mandrikov, Marc Hoffmann

#JPoint #TDD #Bytecode

Moscow, 06.04.2019

Who we are?

Who we are?

Team behind #JaCoCo project

Who we are?

Team behind #JaCoCo project

- Marc Hoffmann, DE/CH, @marcandsweep

Who we are?

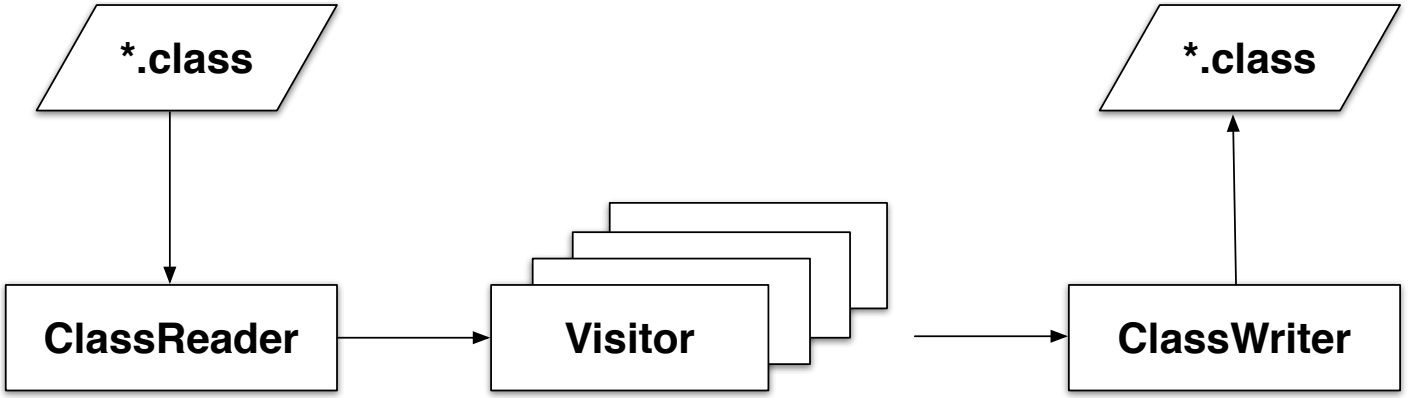
Team behind #JaCoCo project

- Marc Hoffmann, DE/CH, @marcandsweep
- Evgeny Mandrikov, RU/FR, @_Godin_

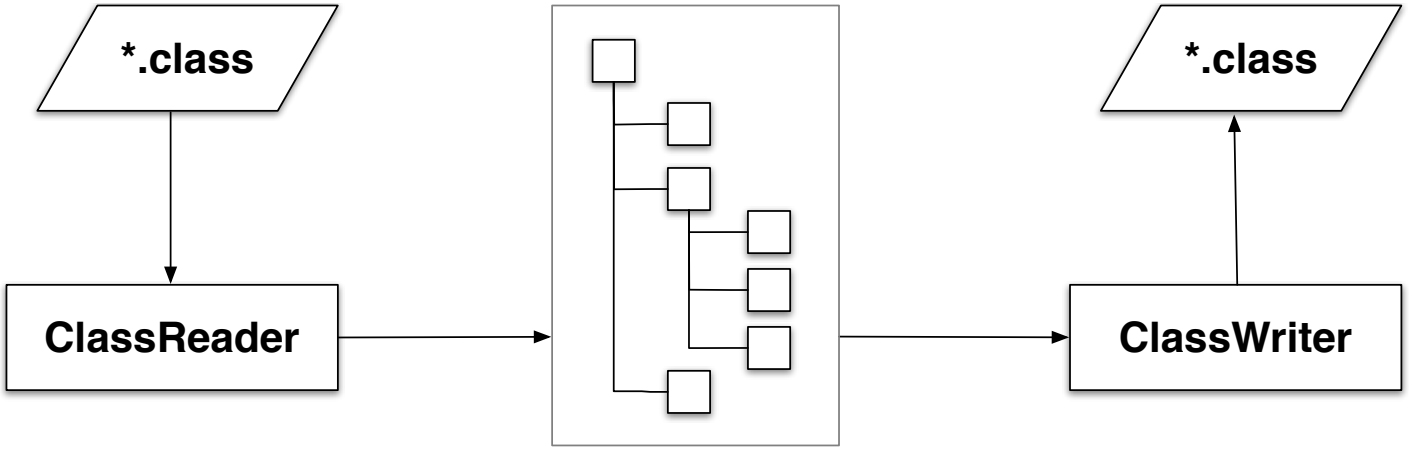
Use Cases for Byte Code Engineering

Example	Read	Write
Compilers	•	•
Scripting Engines		•
Static Analysis	•	
Dynamic Analysis	•	•
Reverse Engineering	•	

ASM - A Bytecode Manipulation Library



ASM - A Bytecode Manipulation Library



Test Driven Bytecode Engineering

Creating or manipulating Java bytecode can be tricky when working with low-level libraries like ASM. Writing and maintaining tools on bytecode level should therefore always be guided by comprehensive tests.

Generation

How to test class creation?

Stack Frames

Data stack for method execution:

- Operand stack (push/pop)
- Local variables (indexed access)

Stack Frames

Data stack for method execution:

- Operand stack (push/pop)
- Local variables (indexed access)

Fixed, predefined stack sizes:

- Defined in class files
- Checked by verifier

What if exception happens in generated code?

How to catch exception in generated code?

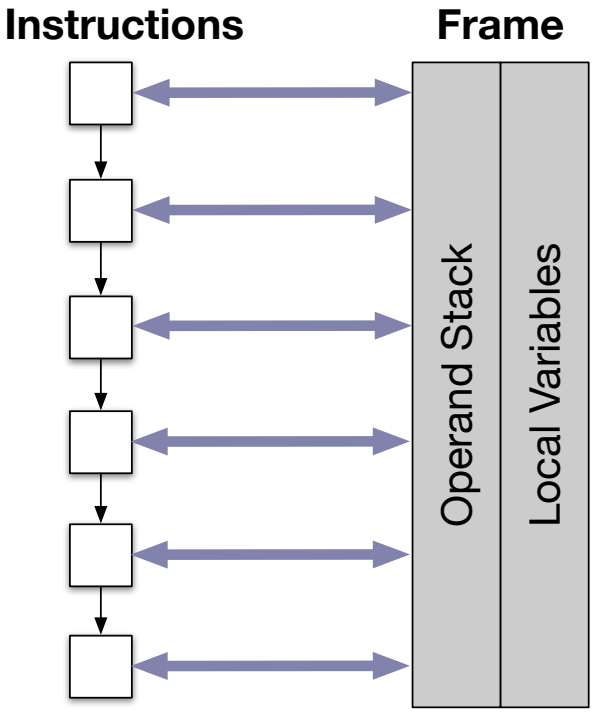
Stack Map Frames

Stack Map Frames

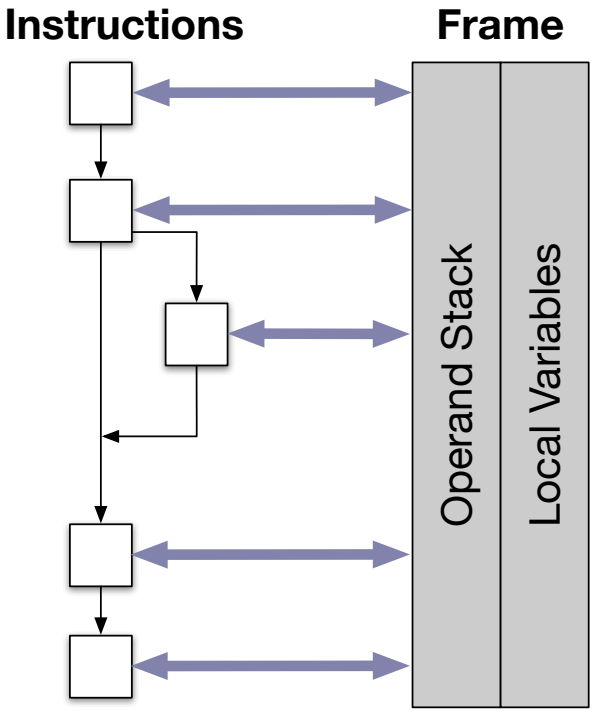
Java Virtual Machine Specification:

> A class file whose version number is 50.0 or above must be verified using the type checking rules given in this section.
>
> The type checker requires a list of stack map frames for each method with a Code attribute.

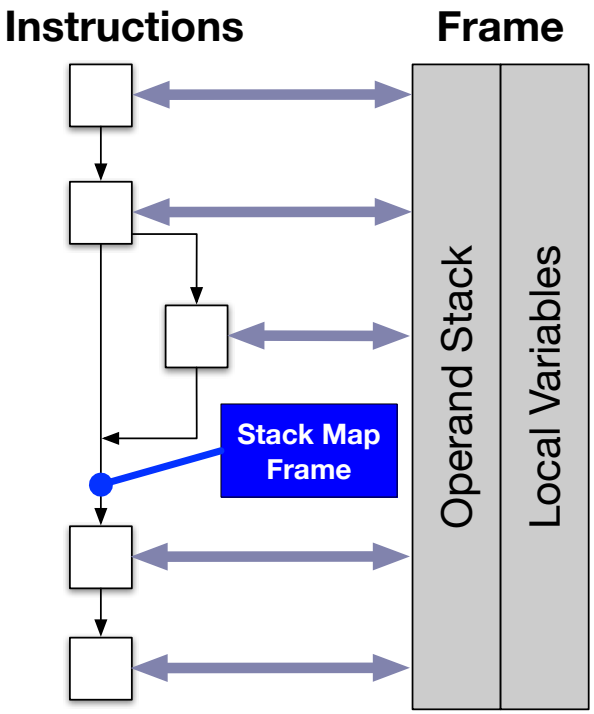
Stack Map Frames



Stack Map Frames



Stack Map Frames



[Java byte-code verification by Nikita Lipsky at JPoint 2017]

Why not always COMPUTE_FRAMES?

- Class hierarchy required to calculate stack map frames from scratch
- Parent types might not (yet) be available
- Loading parent types might cause undesired sideffects

Incremental frames updates

- `asm.ClassVisitor.visitFrame` to adjust existing frames without recalculation
- `asm.AnalyzerAdapter` to insert new ones

Analysis

Analysis

Example: Count executable source lines in a given class

Analysis

Executable Comments

```
class JaCoCoTarget {
  static void main(String[] args) {
    missedBranch(true);
  }

  static void missedBranch(boolean f) {
    if (f) { // assertCovered(1, 1)
      nop(); // assertCovered()
    } else {
      nop(); // assertNotCovered()
    }
  }
}
```

The JDK May Play Tricks on You

The JDK May Play Tricks on You

- VM behaviour depends on class file version

The JDK May Play Tricks on You

- VM behaviour depends on class file version
- VM executes invalid class files (JDK-815718)

The JDK May Play Tricks on You

- VM behaviour depends on class file version
- VM executes invalid class files (JDK-815718)
- javac produces inconsistent class files (JDK-8160928)

The JDK May Play Tricks on You

- VM behaviour depends on class file version
- VM executes invalid class files (JDK-815718)
- javac produces inconsistent class files (JDK-8160928)
- VM might crash on valid class files (JDK-8216970)

Lessons Learned for Bytecode Engineering

Lessons Learned for Bytecode Engineering

- Compiler, JVM, ASM and Spec may have different ideas about valid bytecode.

Lessons Learned for Bytecode Engineering

- Compiler, JVM, ASM and Spec may have different ideas about valid bytecode.
- Implementations and semantic of bytecode may change with classfile versions.

Lessons Learned for Bytecode Engineering

- Compiler, JVM, ASM and Spec may have different ideas about valid bytecode.
- Implementations and semantic of bytecode may change with classfile versions.
- You will see creepy error messages by JVM

Lessons Learned for Bytecode Engineering

- Compiler, JVM, ASM and Spec may have different ideas about valid bytecode.
- Implementations and semantic of bytecode may change with classfile versions.
- You will see creepy error messages by JVM
- Test-first significantly speeds-up development cycles.

Lessons Learned for Bytecode Engineering

- Compiler, JVM, ASM and Spec may have different ideas about valid bytecode.
- Implementations and semantic of bytecode may change with classfile versions.
- You will see creepy error messages by JVM
- Test-first significantly speeds-up development cycles.
- Invest in maintainable and efficient test setups.

Thank you!

- https://github.com/marchof/cafebabe

- Marc Hoffmann, DE/CH, @marcandsweep
- Evgeny Mandrikov, RU/FR, @_Godin_