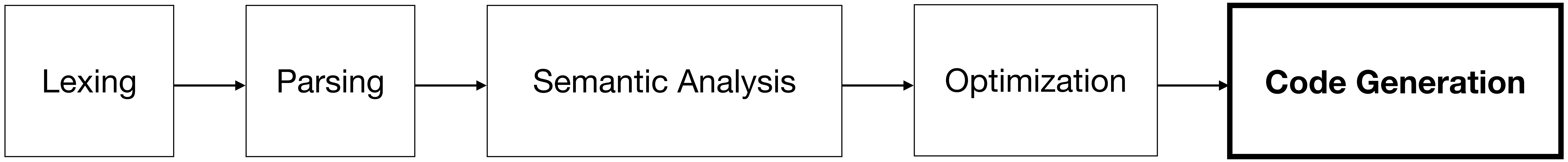
Languages & Translators

LINGI2132

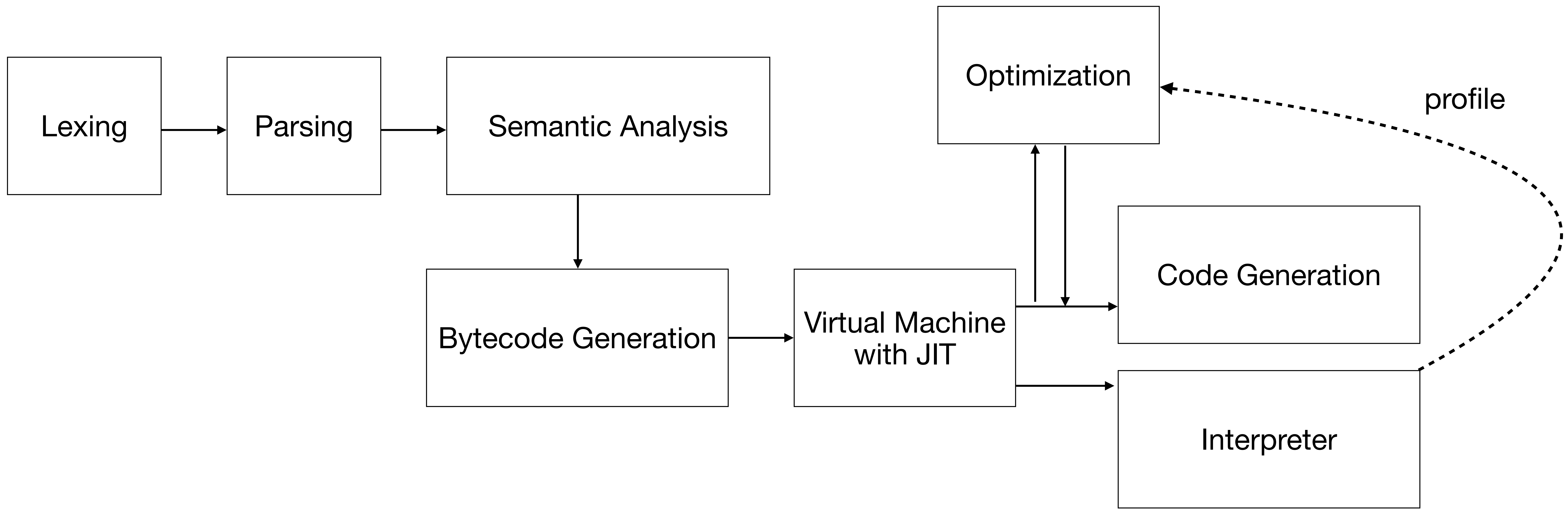
JVM Bytecode

Nicolas LAURENT
Université catholique de Louvain


Lexing
Parsing
Semantic Analysis
Optimization
Code Generation

or…

e.g. Java (JVM) , Javascript (V8), TruffleRuby, Python (PyPy)



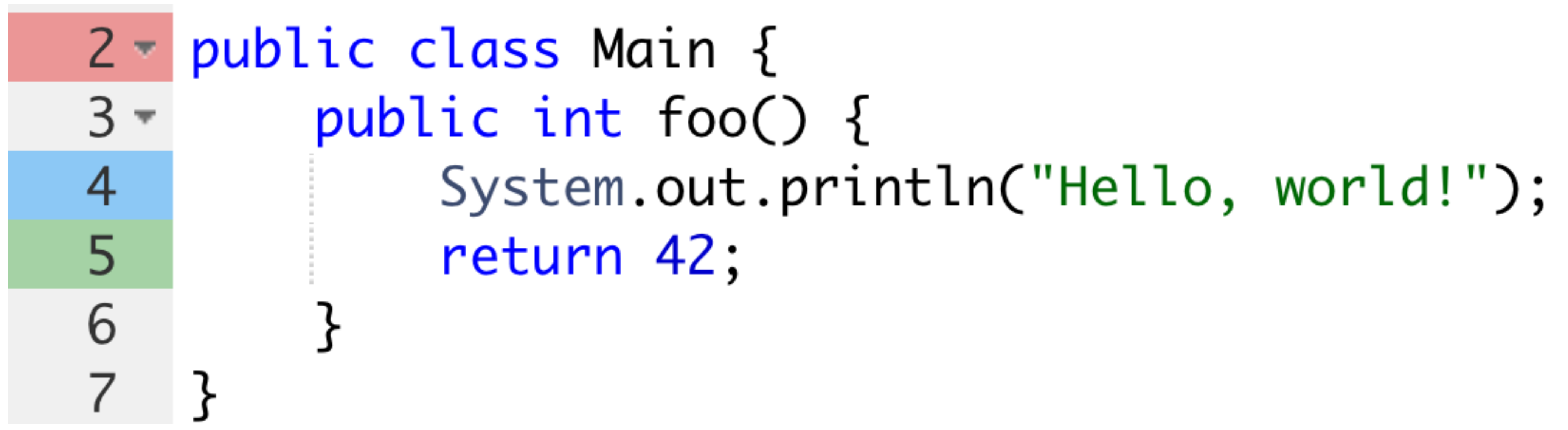
Code Generation

- Compiling by translation
    - Into machine code (e.g. x64) / machine-code-like IR (e.g. LLVM)
        - Needs manual optimization
        - Huge, lots of legacy (x64 > 1000)
        - Lowest-level: more control, but also more work
    - Into byte code
        - Typically higher-level
        - Typically smaller (JVM ~ 200)
    - Into source code (e.g. JavaPoet)
        - Potentially the simplest, but more limited
        - Potentially the slowest (duplicated work)

Why JVM Bytecode?

- Best effort/result trade-off
- Built-in garbage collector / Not good for low-level languages
- Built-in JIT compiler
    - No "need" for bytecode optimization (but possible, e.g. ProGuard)
    - Per-CPU instruction selection
- Built-in polymorphism (Java-style virtual + interface dispatch)
    - more flexibility possible with `invokedynamic`
- Ecosystem: call to/from Java
    - In machine code: call to/from C
- Can map line numbers to instructions
- Portable (LLVM too)
- Requires JVM? jlink, jpackage (+ ProGuard?), GraalVM native-image

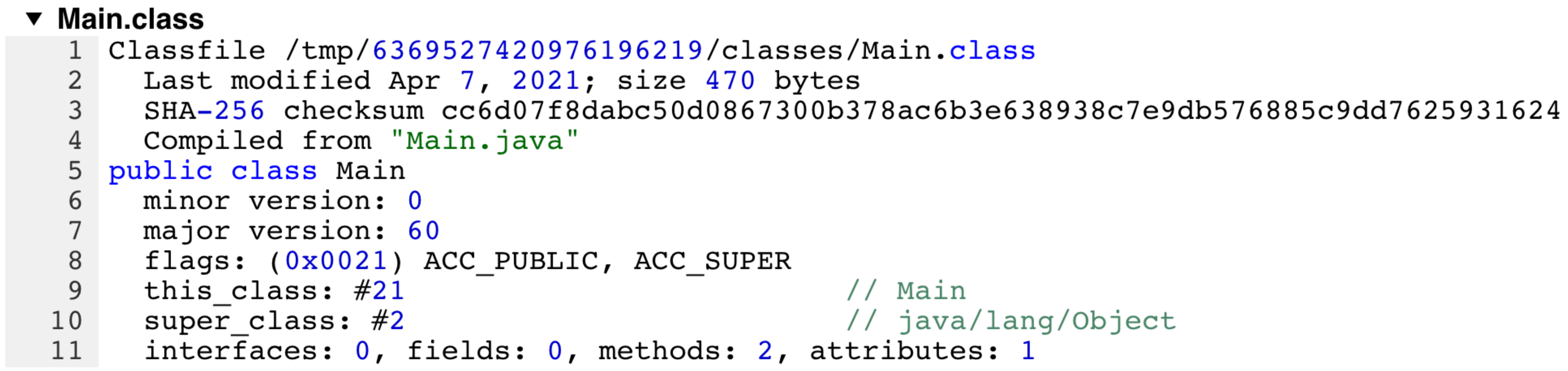
Java to Bytecode



```java
public class Main {
    public int foo() {
        System.out.println("Hello, world!");
        return 42;
    }
}
```

```
> javac Main.java
> javap -v -p Main.class
```

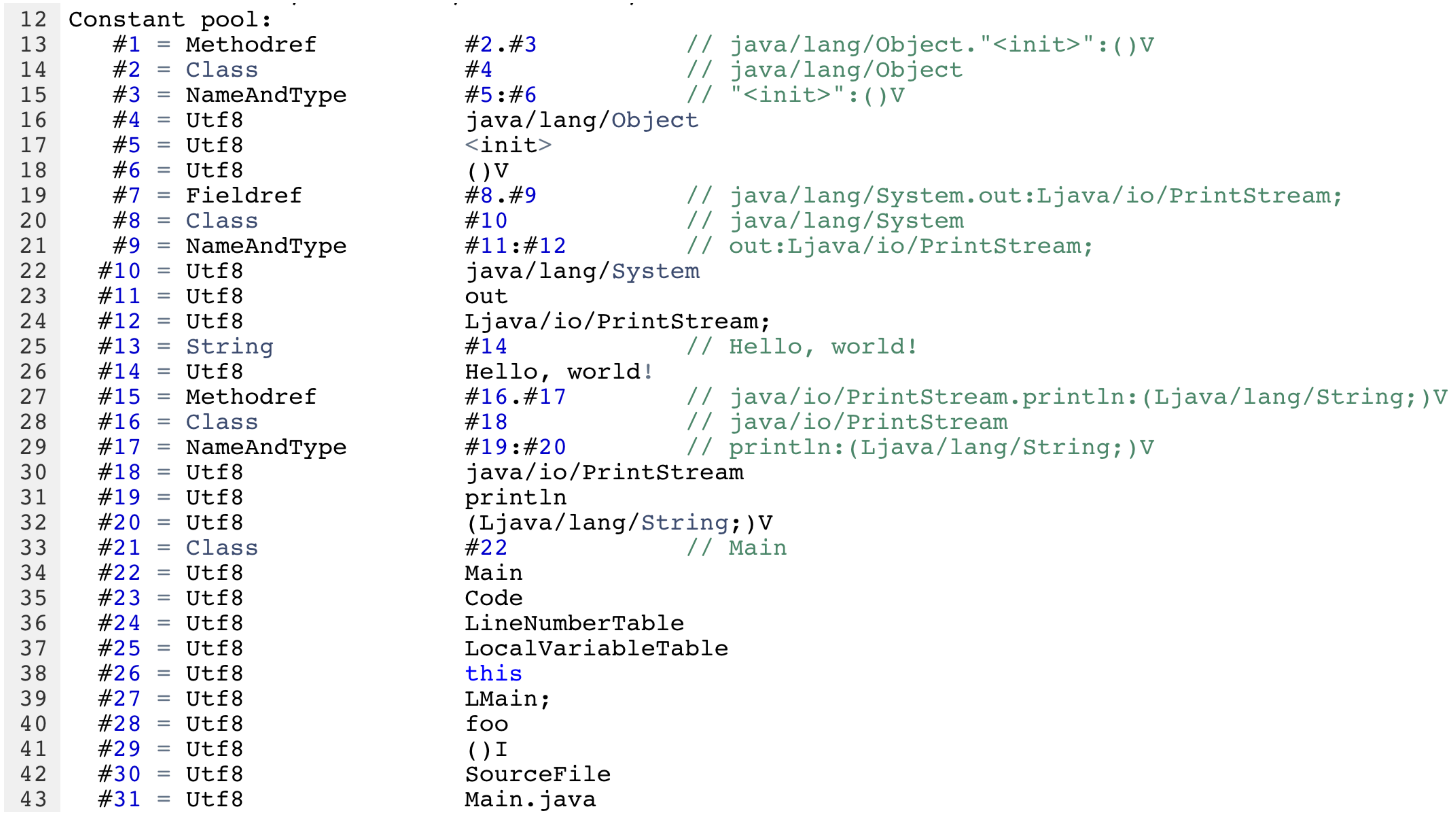
or use https://javap.yawk.at



▼ **Main.class**

```
Classfile /tmp/6369527420976196219/classes/Main.class
  Last modified Apr 7, 2021; size 470 bytes
  SHA-256 checksum cc6d07f8dabc50d0867300b378ac6b3e638938c7e9db576885c9dd7625931624
  Compiled from "Main.java"
public class Main
  minor version: 0
  major version: 60
  flags: (0x0021) ACC_PUBLIC, ACC_SUPER
  this_class: #21                          // Main
  super_class: #2                          // java/lang/Object
  interfaces: 0, fields: 0, methods: 2, attributes: 1
```

Constant Pool



```
Constant pool:
   #1 = Methodref          #2.#3          // java/lang/Object."<init>":()V
   #2 = Class              #4             // java/lang/Object
   #3 = NameAndType        #5:#6          // "<init>":()V
   #4 = Utf8               java/lang/Object
   #5 = Utf8               <init>
   #6 = Utf8               ()V
   #7 = Fieldref           #8.#9          // java/lang/System.out:Ljava/io/PrintStream;
   #8 = Class              #10            // java/lang/System
   #9 = NameAndType        #11:#12        // out:Ljava/io/PrintStream;
  #10 = Utf8               java/lang/System
  #11 = Utf8               out
  #12 = Utf8               Ljava/io/PrintStream;
  #13 = String             #14            // Hello, world!
  #14 = Utf8               Hello, world!
  #15 = Methodref          #16.#17        // java/io/PrintStream.println:(Ljava/lang/String;)V
  #16 = Class              #18            // java/io/PrintStream
  #17 = NameAndType        #19:#20        // println:(Ljava/lang/String;)V
  #18 = Utf8               java/io/PrintStream
  #19 = Utf8               println
  #20 = Utf8               (Ljava/lang/String;)V
  #21 = Class              #22            // Main
  #22 = Utf8               Main
  #23 = Utf8               Code
  #24 = Utf8               LineNumberTable
  #25 = Utf8               LocalVariableTable
  #26 = Utf8               this
  #27 = Utf8               LMain;
  #28 = Utf8               foo
  #29 = Utf8               ()I
  #30 = Utf8               SourceFile
  #31 = Utf8               Main.java
```

Constructor

```
{
  public Main();
    descriptor: ()V
    flags: (0x0001) ACC_PUBLIC
    Code:
      stack=1, locals=1, args_size=1
        start local 0 // Main this
         0: aload_0
         1: invokespecial #1                  // Method java/lang/Object."<init>":()V
         4: return
        end local 0 // Main this
      LineNumberTable:
        line 2: 0
      LocalVariableTable:
        Start  Length  Slot  Name   Signature
            0       5     0  this   LMain;
```

Method foo

```
public int foo();
  descriptor: ()I
  flags: (0x0001) ACC_PUBLIC
  Code:
    stack=2, locals=1, args_size=1
      start local 0 // Main this
       0: getstatic     #7                  // Field java/lang/System.out:Ljava/io/PrintStream;
       3: ldc           #13                 // String Hello, world!
       5: invokevirtual #15                 // Method java/io/PrintStream.println:(Ljava/lang/String;)V
       8: bipush        42
      10: ireturn
      end local 0 // Main this
    LineNumberTable:
      line 4: 0
      line 5: 8
    LocalVariableTable:
      Start  Length  Slot  Name   Signature
          0      11     0  this   LMain;
```

Next Time
JVM Bytecode Instructions