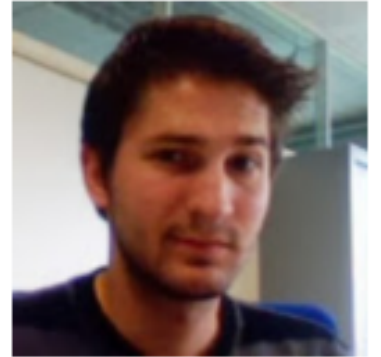


# Byte Buddy : **bytecode** **gen made *easy* !**

# About me

🐦 @NicolasComet

🐙 <https://github.com/ncomet>



- Developer since 7 years
- Software R&D Engineer @Lectra, worldwide leader in integrated softwares, machines and cloud services for soft material industries
- Speaker & Member of Bordeaux JUG





FASHION AND APPAREL

AUTOMOTIVE

FURNITURE

OTHER INDUSTRIES

CUSTOMER STORIES

Search



ABOUT LECTRA

Home — Careers

CAREERS

## Careers

JOIN AN  
INTERNATIONAL  
COMPANY

Global leader in most of its markets, Lectra is a transnational group with nearly 1,500 professionals employed in more than 100 countries. Thanks to this unparalleled international network and the expertise of its teams, Lectra is a privileged partner of major



# How it all began

---

*"Making Java more dynamic"* @Devoxx France 2015

<https://youtu.be/vjv4idwQL7k>

## Rafael Winterhalter



- Non business/domain related
- Orthogonal preoccupations









# Cross Cutting Concerns

---

- Non business/domain related
- Orthogonal preoccupations



- Logging 
- Caching 
- Monitoring 
- Data validation 
- Real time constraints 
- Persistence 
- Transactions 
- ...

- Transparent
- Non intrusive
- Strongly typed
- Regardless of type
- Lightweight



# (Some) solutions

---

- Reflection
- Aspect Oriented Programming (AOP)
- Bytecode generation

`java.lang.reflect`

Reading Type metamodel at runtime

Calling constructors, methods, access attributes (sometimes unsafely)

`java.lang.reflect`

Reading Type metamodel at runtime

Calling constructors, methods, access attributes (sometimes unsafely)

Mostly about *Introspection*

`java.lang.reflect`

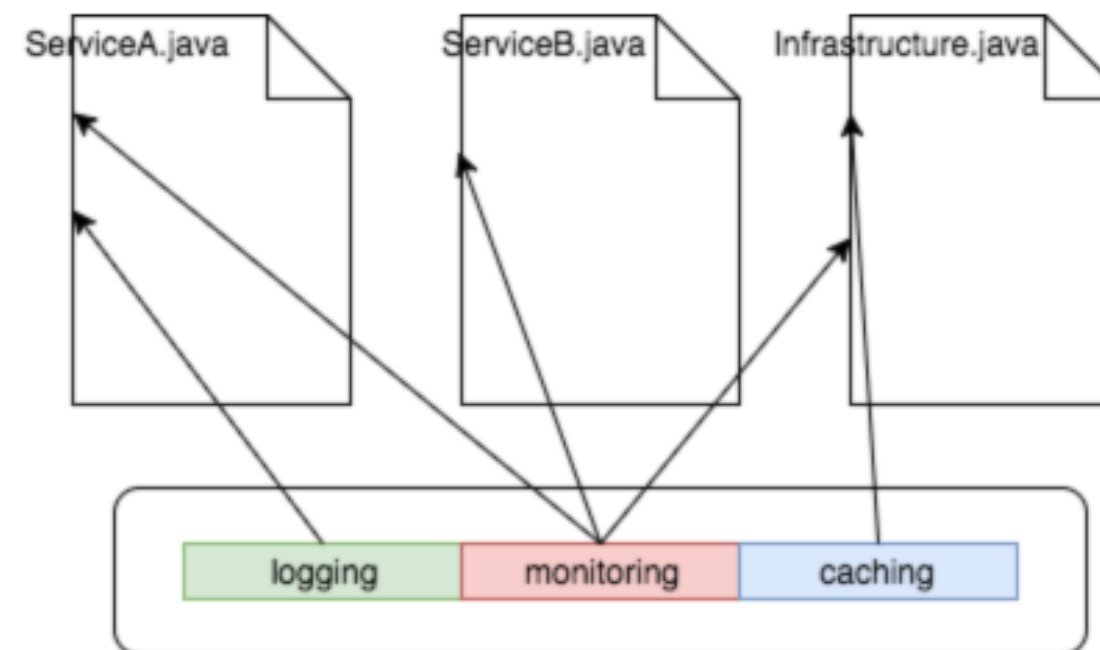
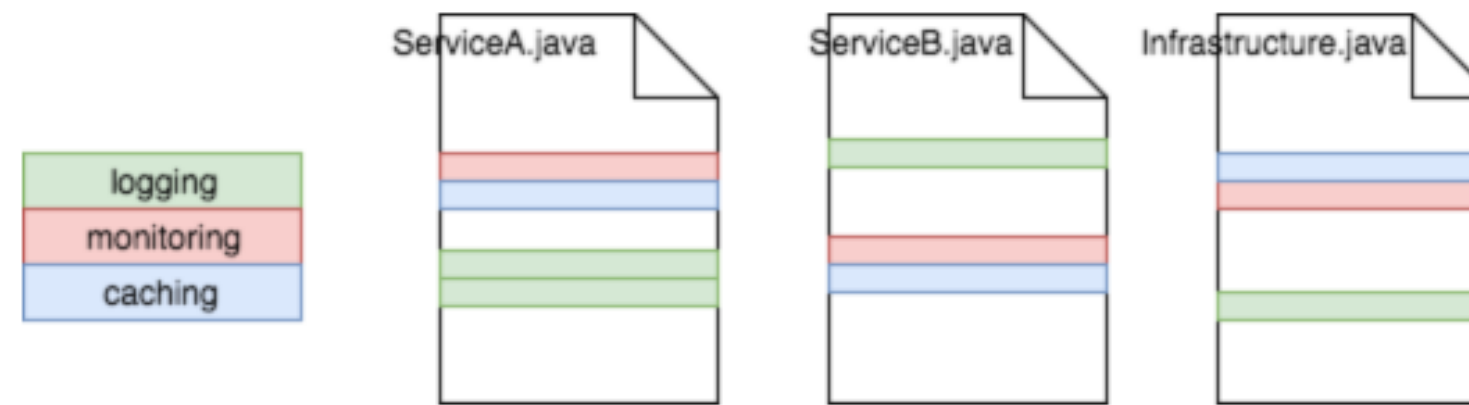
Reading Type metamodel at runtime

Calling constructors, methods, access attributes (sometimes unsafely)

Mostly about *Introspection*

It has a cost (JIT is useless)

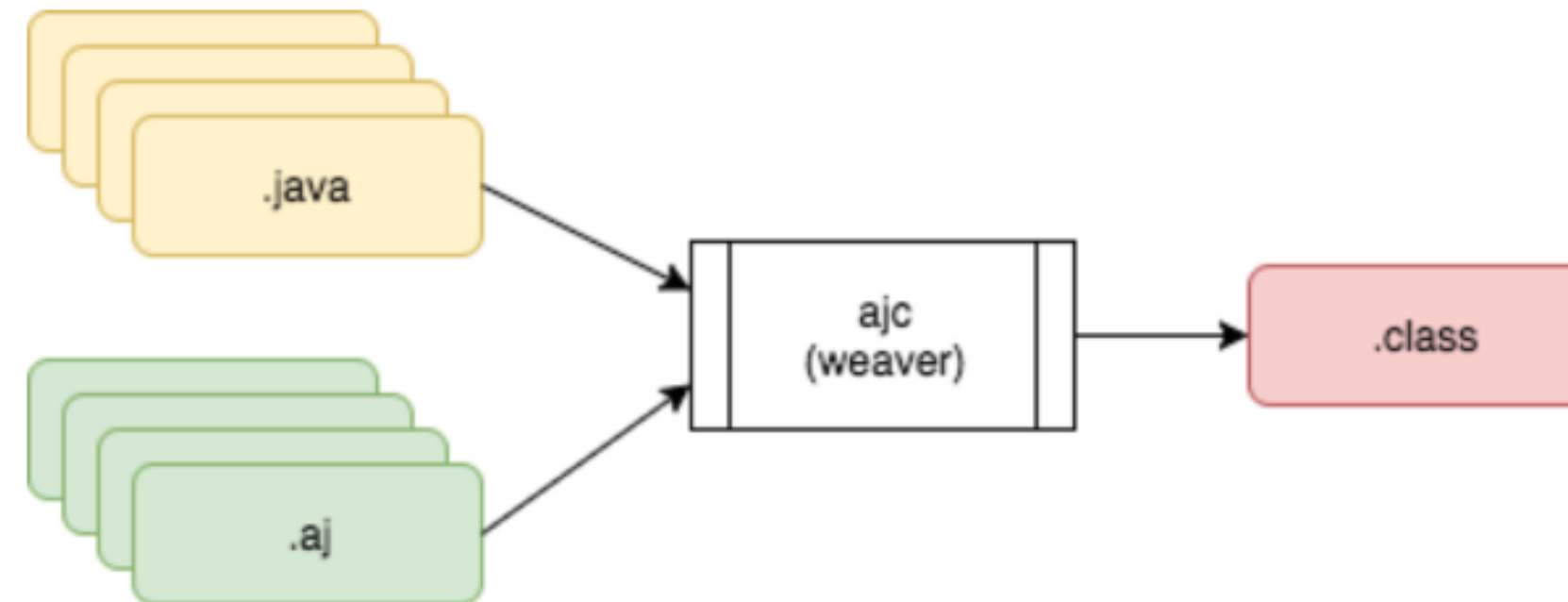
# AOP concepts





- **Aspect** Description of a cross cutting concern
- **Join point** A point during the execution of code (method execution, attribute access)
- **Advice** Action taken by an aspect at a particular join point
- **Pointcut** A regular expression that matches join points.

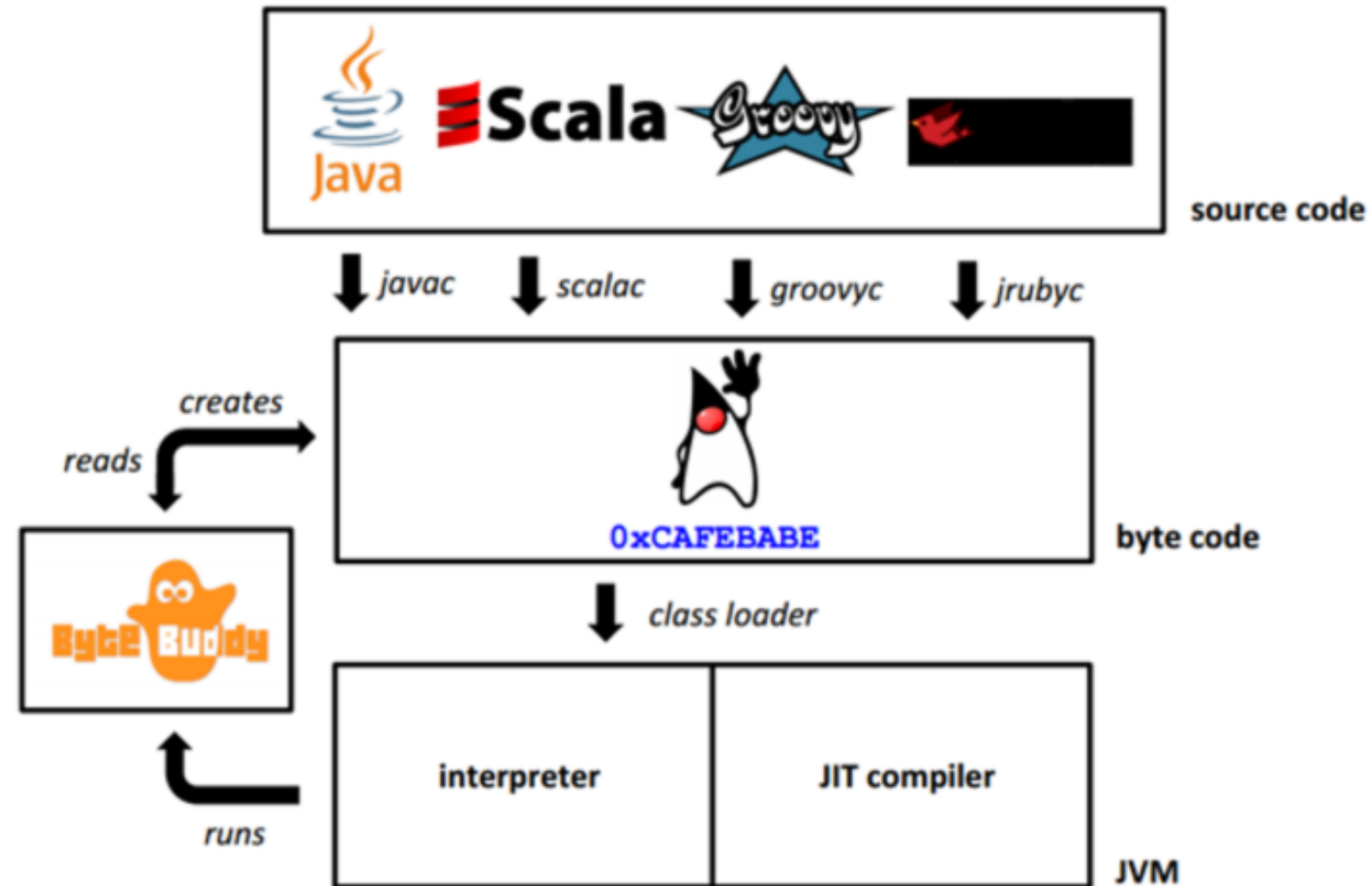
- **Aspect** Description of a cross cutting concern
- **Join point** A point during the execution of code (method execution, attribute access)
- **Advice** Action taken by an aspect at a particular join point
- **Pointcut** A regular expression that matches join points.



```
// access flags 0x1
public getName()Ljava/lang/String;
L0
  LINENUMBER 16 L0
  ALOAD 0
  GETFIELD domain/Cat.name : Ljava/lang/String;
  ARETURN
L1
  LOCALVARIABLE this Ldomain/Cat; L0 L1 0
  MAXSTACK = 1
  MAXLOCALS = 1

// access flags 0x1
public setName(Ljava/lang/String;)V
L0
  LINENUMBER 20 L0
  ALOAD 0
  ALOAD 1
  PUTFIELD domain/Cat.name : Ljava/lang/String;
L1
  LINENUMBER 21 L1
  RETURN
L2
  LOCALVARIABLE this Ldomain/Cat; L0 L2 0
  LOCALVARIABLE name Ljava/lang/String; L0 L2 1
  MAXSTACK = 2
  MAXLOCALS = 2
```

# Bytecode generation



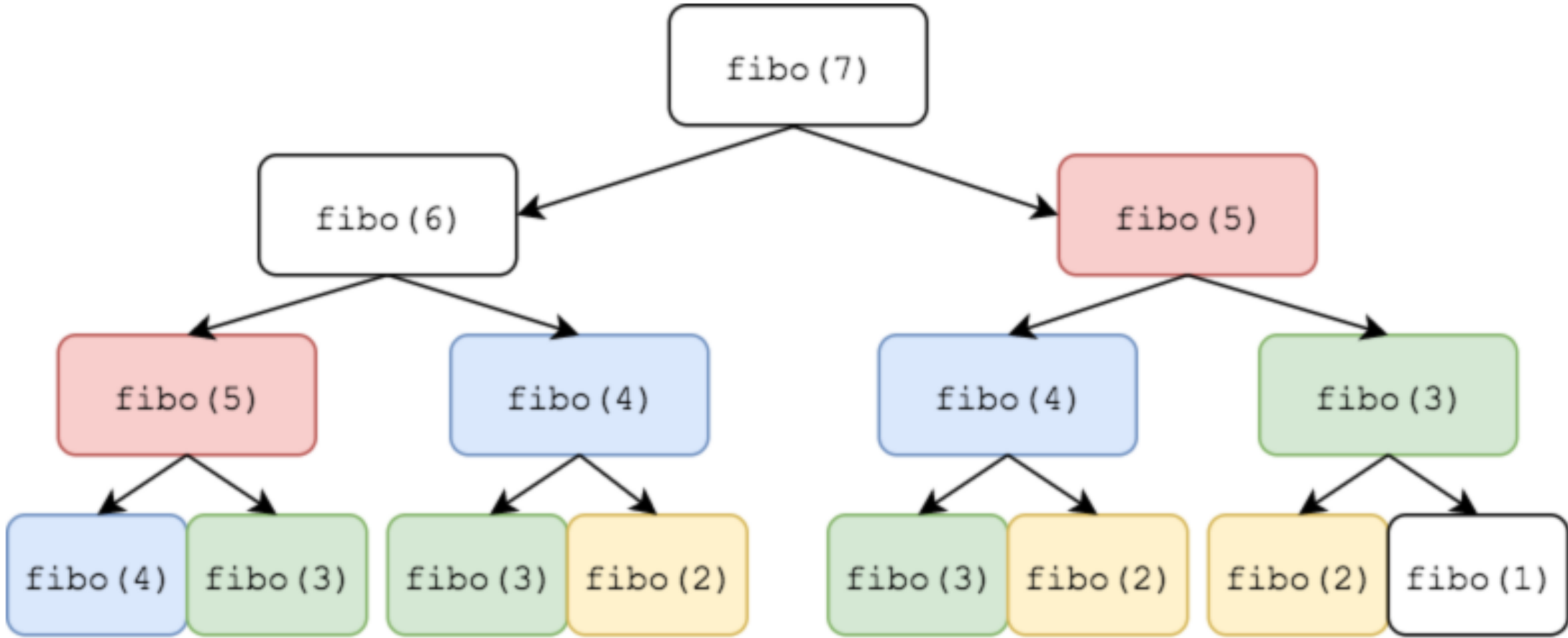
The Spring logo features the word "spring" in a green, lowercase, sans-serif font, with a small green leaf icon positioned above the letter "i".The Hibernate logo consists of a small, stylized hexagonal icon to the left of the word "HIBERNATE" in a blue, uppercase, sans-serif font.The Mockito logo features the word "mockito" in a green, lowercase, sans-serif font, followed by a small icon of a glass with a straw and a green leaf.The Guice logo displays the word "Guice" in a blue, uppercase, sans-serif font, with the letter "u" in a lighter blue color.The EclipseLink logo features the word "eclipse" in a purple, lowercase, sans-serif font, followed by a stylized purple arc and the word "link" in a blue, uppercase, sans-serif font.The Play logo consists of the word "play" in a black, lowercase, sans-serif font, followed by a green right-pointing triangle.The Clover logo features a blue four-leaf clover icon to the left of the word "Clover" in a blue, uppercase, sans-serif font.The OpenEJB logo consists of a red and black icon to the left of the word "OpenEJB" in a red, uppercase, sans-serif font.The Apache Wicket logo features an orange circle containing a white crown icon, with the text "APACHEWICKET" in a small, black, uppercase, sans-serif font below it.The Apache Shiro logo features a blue and white icon to the left of the word "SHIRO" in a blue, uppercase, sans-serif font, with "APACHE" in a smaller font above it.The OpenJDK logo consists of the word "OpenJDK" in a blue, uppercase, sans-serif font, with the letter "O" in a lighter blue color.The Grails logo features a green circle containing a white icon of a person with arms raised, followed by the word "GRAILS" in a black, uppercase, sans-serif font.



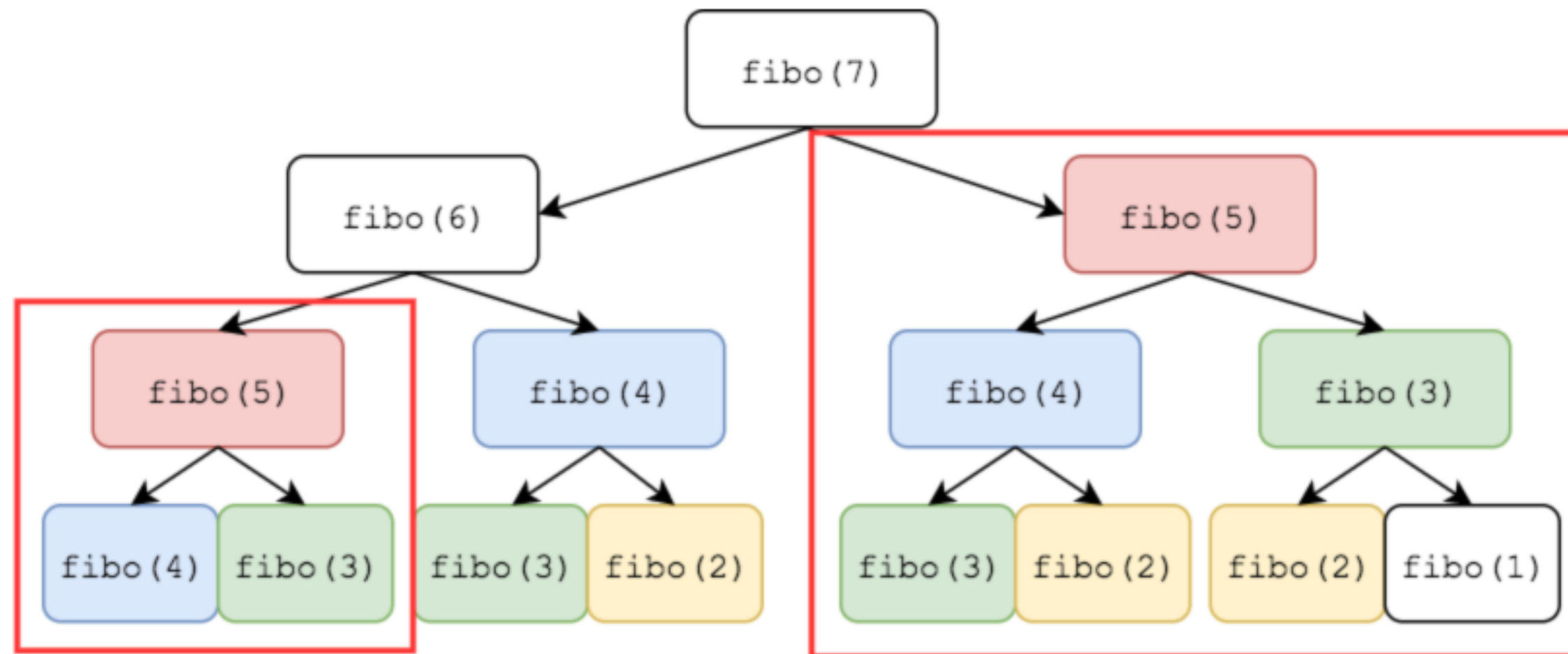
- $n \in \mathbb{N}$

$$f(n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ f(n-1) + f(n-2) & \text{if } n > 1 \end{cases}$$

- Call tree



- Memoization



# Some code !

---



Calling `fibonacci(42)` (average results)

| Version                | Time                               |
|------------------------|------------------------------------|
| Raw Fibonacci          | 1123.658 ms                        |
| AspectJ (compile time) | 0.013 ms                           |
| Byte Buddy (runtime)   | 0.689 ms                           |
| Spring AOP             | 2123 ms (first time, then instant) |




- AspectJ
  - compile time weaving (ajc)
  - post-compile weaving (on classes and jars)
  - load time weaving (agent)
  - intercept everything
- Spring AOP
  - proxy-based
    - Interface → Java dynamic proxy
    - else CGLIB bytecode generated proxy
  - good AspectJ integration if you need more


- AspectJ
  - - Setup
  - - DSL to learn
  - + Performance
  - + Non intrusive
  - + Span
- Spring AOP
  - - Not really AOP
  - - Component's public methods only
  - - / + Framework
  - + Spring integration
  - + Migration to AspectJ
- Byte Buddy
  - - / + No compile time
  - + Library
  - + Java DSL API
  - + Performance
  - + Agent writing help

# Byte Buddy

Open Source (license Apache), used by Mockito, Hibernate, Google Bazle, and others

 Unwatch ▼

108

 Unstar

1,667

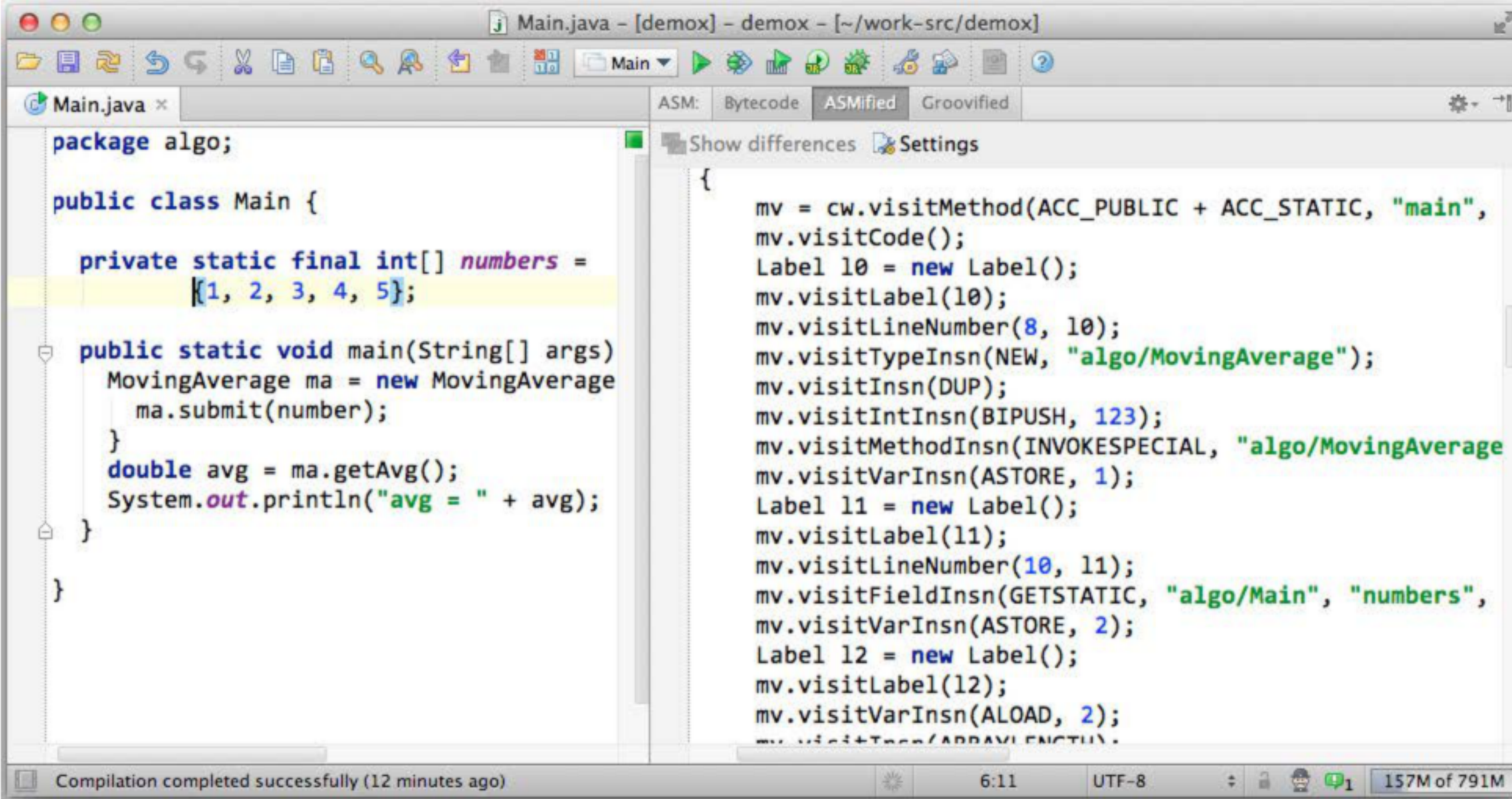
 Fork

180

 <https://github.com/raphw/byte-buddy>

 <http://bytebuddy.net>

- Light
- Easy to use (compared to CGLIB, BCEL, ASM)
- Become a library writer



The screenshot shows an IDE window titled "Main.java - [demox] - demox - [~/work-src/demox]". The left pane displays the Java source code for a class named `Main` in the `algo` package. The right pane shows the corresponding ASM code, which is a sequence of visitor method calls. The status bar at the bottom indicates "Compilation completed successfully (12 minutes ago)", the time is 6:11, the encoding is UTF-8, and the file size is 157M of 791M.

```
package algo;

public class Main {

    private static final int[] numbers =
        {1, 2, 3, 4, 5};

    public static void main(String[] args)
    {
        MovingAverage ma = new MovingAverage
        {
            ma.submit(number);
        };
        double avg = ma.getAvg();
        System.out.println("avg = " + avg);
    }
}
```

```
{
    mv = cw.visitMethod(ACC_PUBLIC + ACC_STATIC, "main",
    mv.visitCode();
    Label l0 = new Label();
    mv.visitLabel(l0);
    mv.visitLineNumber(8, l0);
    mv.visitTypeInsn(NEW, "algo/MovingAverage");
    mv.visitInsn(DUP);
    mv.visitIntInsn(BIPUSH, 123);
    mv.visitMethodInsn(INVOKESTATIC, "algo/MovingAverage
    mv.visitVarInsn(ASTORE, 1);
    Label l1 = new Label();
    mv.visitLabel(l1);
    mv.visitLineNumber(10, l1);
    mv.visitFieldInsn(GETSTATIC, "algo/Main", "numbers",
    mv.visitVarInsn(ASTORE, 2);
    Label l2 = new Label();
    mv.visitLabel(l2);
    mv.visitVarInsn(ALOAD, 2);
    mv.visitInsn(ARRAYLENGTH);
}
```



Slides :

 <https://ncomet.github.io/javaone2017-bytebuddy/bytebuddy.html>

Sources :

 <https://github.com/ncomet/javaone2017-bytebuddy>



- Adding behavior
  - AOP → Implementing multiple cross cutting concerns
  - Byte Buddy → Writing libraries/frameworks or agents
- Discovering at runtime
  - Reflection → Custom serialization, *nasty* things (setting private fields...)