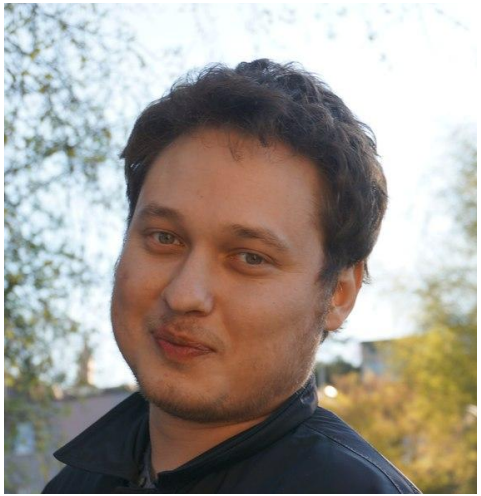
Java Code Coverage Mechanics

Evgeny Mandrikov
Marc Hoffmann
#JokerConf 2017, Saint-Petersburg

Evgeny Mandrikov



@_Godin_

Godin

Marc Hoffmann

@marcandsweep

marchof

JaCoCo and **Eclipse EclEmma** Project Leads

```
/* TODO Don't forget to add huge disclaimer that
 * all opinions hereinbelow are our own and not
 * our employer's.
 *
 * They can only dream that they own them.
 */
```

Java **Code Coverage** Mechanics

Java Code Coverage Mechanics

Java Code Coverage Mechanics

Java
Code
Coverage
Mechanics

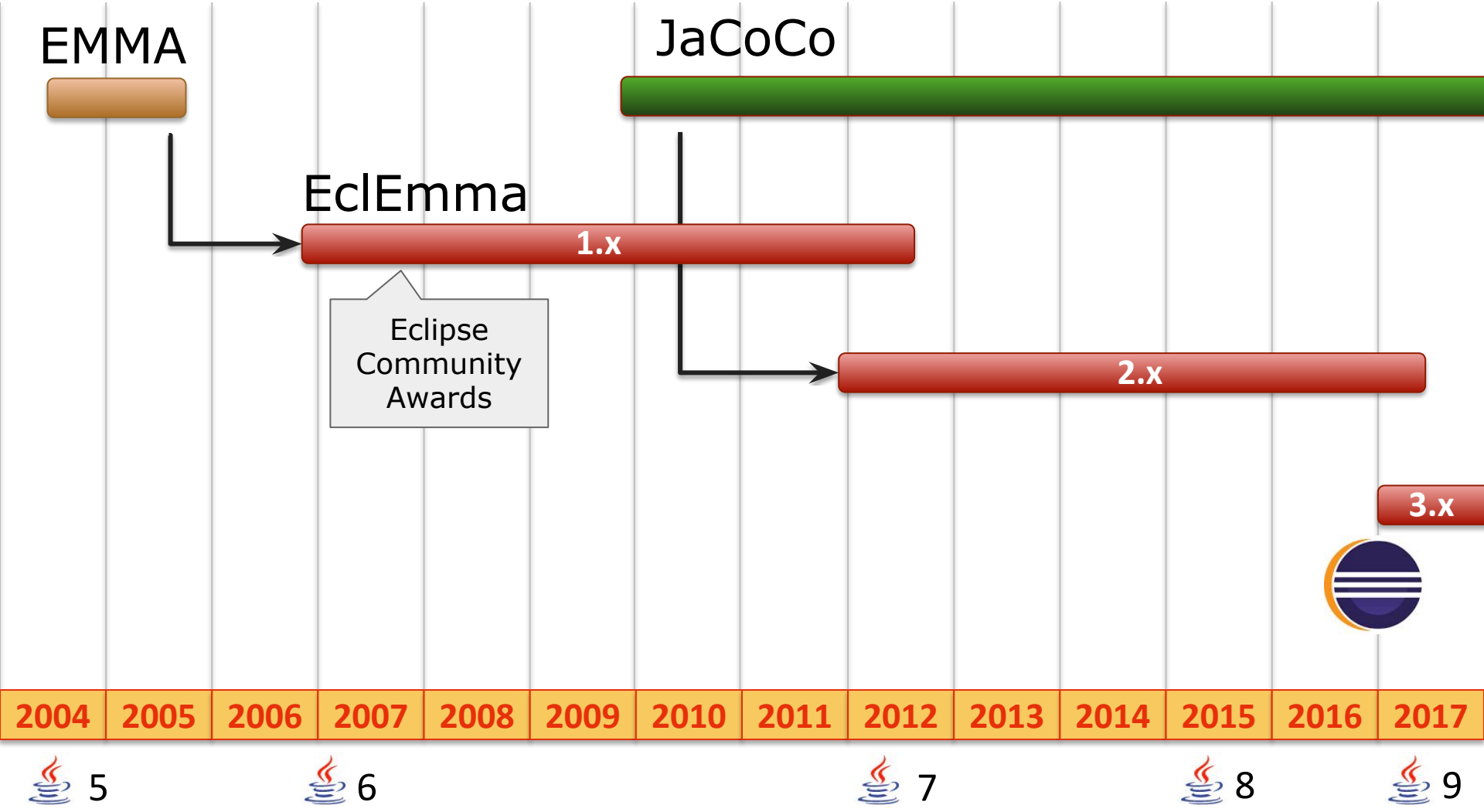
Real Disclaimer

Blood and guts of JVM

Strong language - bytecode

Intense violence for brain

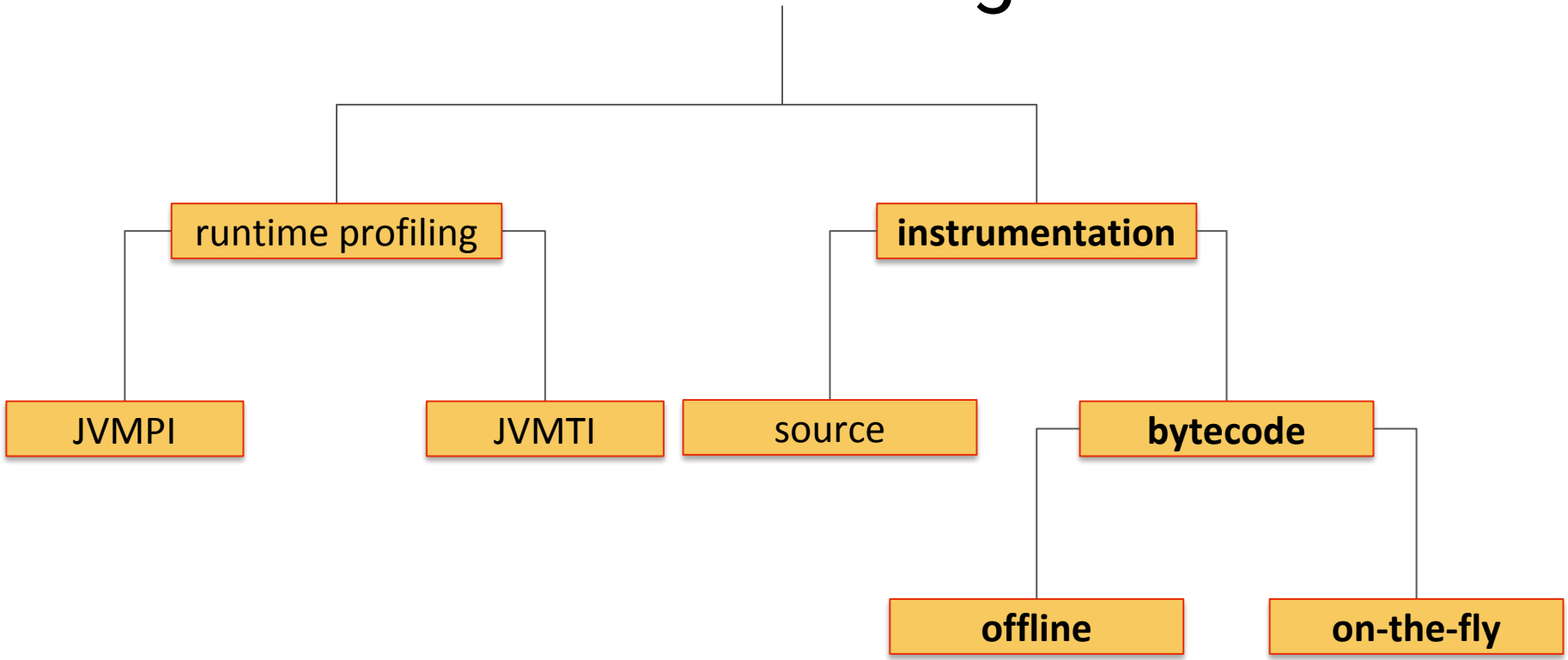
Implementation details!!!


EMMA
JaCoCo
EclEmma
1.x
Eclipse
Community
Awards
2.x
3.x
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
5
6
7
8
9

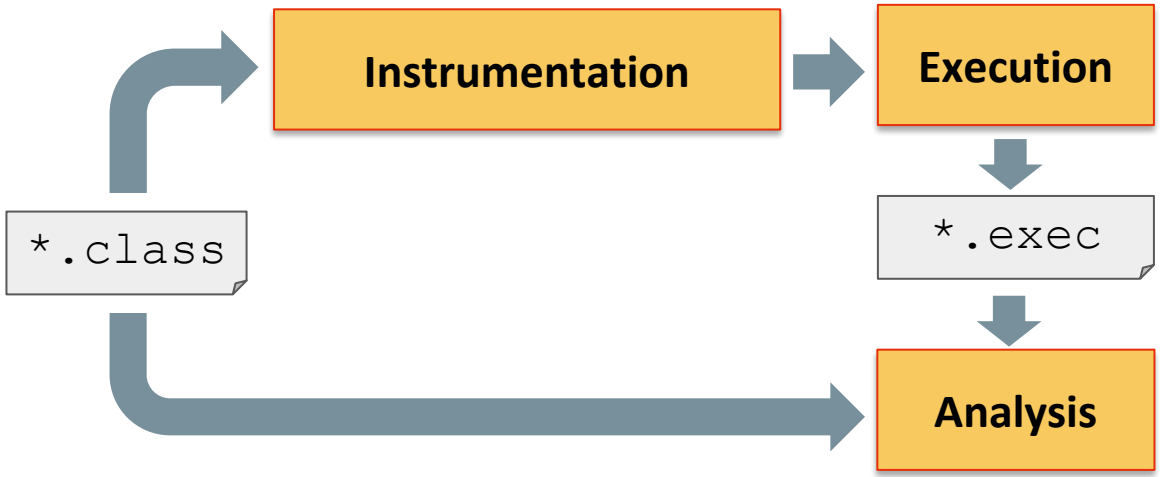
Observer Effect

> In physics, the term observer effect refers to changes that the act of observation will make on a phenomenon being observed. This is often the result of **instruments that**, by necessity, **alter the state of what they measure** in some manner.
>
> Wikipedia

Code Coverage



JaCoCo Works on Class Files



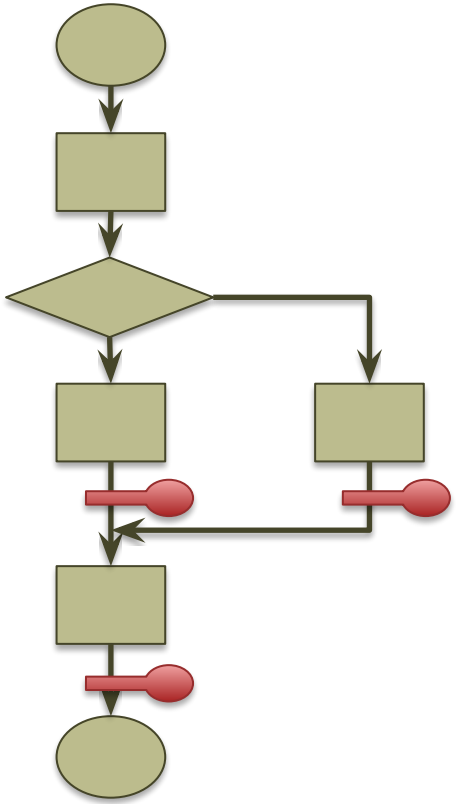

```
120.    public static IRuntime createFor(final Instrumentation
121.            final String className, final String accessFiel
122.            throws ClassNotFoundException {
123.        final ClassFileTransformer transformer = new ClassF
124.            public byte[] transform(final ClassLoader loade
125.                    final String name, final Class<?> class
126.                    final ProtectionDomain protectionDomain
127.                    throws IllegalClassFormatException {
128.                if (name.equals(className)) {
129.                    return instrument(source, accessFieldNa
130.                }
131.                return null;
132.            }
133.        };
134.        inst.addTransformer(transformer);
135.        final Class<?> clazz = Class.forName(className.repl
136.        inst.removeTransformer(transformer);
137.        try {
138.            clazz.getField(accessFieldName);
139.        } catch (final NoSuchFieldException e) {
140.            throw new RuntimeException(format(
141.                    "Class %s could not be instrumented.",
142.        }
143.        return new ModifiedSystemClassRuntime(clazz, access
144.    }
```

Just Set an Arg or the JVM

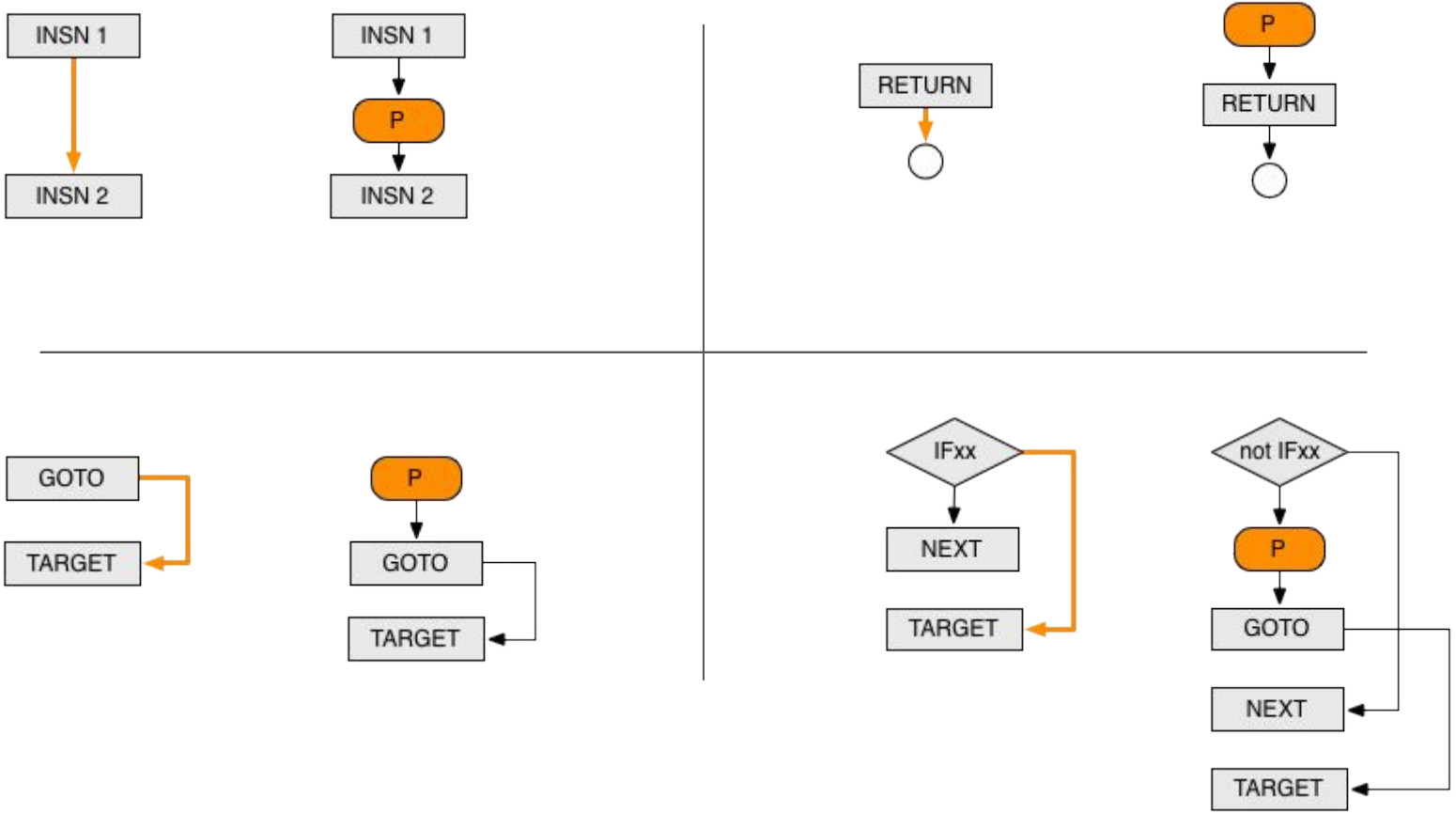
```
java -javaagent:jacocoagent.jar[=options] Application
```

```
class PreMain {
  public static void premain(
    String options,
    Instrumentation instrumentation
  ) throws Exception {
    instrumentation.addTransformer(…);
  }
}
```

Java Byte Code Instrumentation



Probe Strategies



JaCoCo Validation Test Suite

```
// 6. Executed while block
int i = 0;
while (i++ < 3) { // $line-whiletruefalse$
    nop(); // $line-executedwhile$
}

// 7. Executed do while block
do {
    nop(); // $line-executeddowhile$
} while (f());

// 8. Missed for block
for (nop(); f(); nop()) { // $line-missedforincrementer$
    nop(); // $line-missedfor$
}

// 9. Executed for block
for (int j = 0; j < 1; j++) { // $line-executedforincrementer$
    nop(); // $line-executedfor$
}

// 10. Missed for each block
for (Object o : Collections.emptyList()) { // $line-missedforeachincreme
    nop(o); // $line-missedforeach$
}
```

Tested on 5 Major JDK Versions

jacoco / jacoco  build passing

Current Branches Build History Pull Requests > Build #1064

✓ Do not violate JVMS regarding initialization of final fields

Without this change instrumented classes can't pass checks and cause IllegalAccessError starting from OpenJDK 9 EA b127 (see https://bugs.openjdk.java.net/browse/JDK-8157181).

Commit 06f52f0

Compare ba923a0..06f52f0

 Evgeny Mandrikov authored and committed

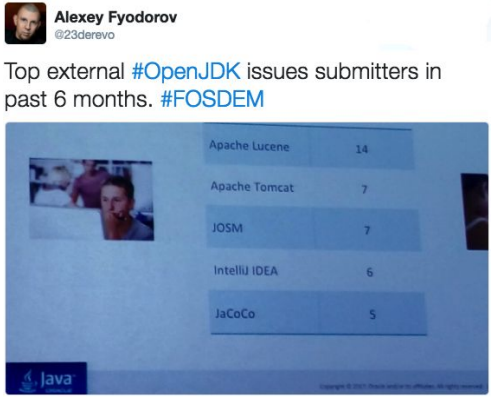
Build Jobs

✓	# 2188.1	JDK=5
✓	# 2188.2	JDK=6
✓	# 2188.3	JDK=7
✓	# 2188.4	JDK=8
✓	# 2188.5	JDK=8 ECJ=true
✓	# 2188.6	JDK=9

Allowed Failures ?

✓	# 2188.7	JDK=8-ea

Not Only Issues in JaCoCo...



T	Key	Summary	Assignee	Reporter	P ↓	Status
Bug	JDK-8154017	Shutdown hooks are racing against shutdown sequence, if System.exit()-calling thread is interrupted	Chris Hegarty	Aleksey Shipilev	P2	CLOSED
Bug	JDK-8164302	No initialization for super interface with default method	Karen Kinnear			
Bug	JDK-8080555	Different bytecode between JDK8u45 and JDK8u60-ea-b12	Vicente Arturo Romero Zaldivar			
Bug	JDK-8134862	JVM crash at PhaseIdealLoop::idom_no_update	Balchandra Vaidya			
Bug	JDK-8073658	Invalid annotations in bridge methods	Srikanth Adayapalam			
Bug	JDK-8131041	Garbage in output of DecimalFormat	Naoto Sato			
Bug	JDK-8163969	Cyclic interface initialization causes JVM crash	Coleen Phillimore			

Details

Type:	Bug
Status:	CLOSED
Priority:	P2
Resolution:	Fixed
Affects Version/s:	7, 8, 9
Fix Version/s:	9
Component/s:	core-libs
Labels:	autoverify foss-libs jacoco-found
Subcomponent:	java.lang
Resolved In Build:	b125
Verification:	Verified


ALWAYS WONDERED... JUST WHICH OF US... WAS FASTEST...

Probe

- Minimal runtime overhead
- No side effects on application code
- Thread safe
- Record execution
- Identification

```
probes[id] = true;
```

```
ALOADx   probes
xPUSH    id
ICONST_1
BASTORE
```

How to get `probes[ ]` ?

```
class Fun {
  void fun() {
    boolean[] probes = … ;  // ???
    probes[0] = true;
    …
    probes[…] = true;
  }
}
```

Let’s Have a Look at Java Assertions

```
class Fun {
  void fun(boolean x) {
    assert x;
  }
}
```

```
// javap -c Fun

  getstatic $assertionsDisabled
  ifne L
  iload_1
  ifne L
  new java/lang/AssertionError
  dup
  invokespecial <init>()V
  athrow
L:
  return
```

Bytecode of Assertions

```
class Fun {
  static final synthetic boolean $assertionsDisabled
    = Fun.class.desiredAssertionStatus();

  void fun(boolean x) {
    if (! $assertionsDisabled)
      if (! x)
        throw new AssertionError();
  }
}
```

Nice Idea!

```
class Fun {
  static final synthetic boolean[] $jacocoData = … ;

  void fun() {
    boolean[] probes = $jacocoData;
    probes[0] = true;
    …
    probes[…] = true;
  }
}
```

Synthetic?

Bridge Methods

```
class Outer {
  private int counter;


  class Inner {
    void inc() {
      counter++;
    }
  }
}
```

```
// javap -v Outer
synthetic static
~~private~~
int access$008(Outer o) {
  return o.counter++;
}

// javap -c Outer$Inner
synthetic final Outer this$0;

void inc() {
  Outer.access$008(this$0)
```

Methods in Enums

```
enum Fun {
  C
}
```

```
// javap -v Fun

~~synthetic~~ static
Fun valueOf(java.lang.String);

~~synthetic~~ static
Fun[] values();
```

Lambdas

```
class Fun {
  void exec(Runnable task) {
    task.run();
  }
  void fun() {
    exec(() -> {
      …
    });
  }
}
```

```
// javap -c -p -l Fun

private
static
synthetic
lambda$fun$0() {
  …
}
```


Enjoy
Hardcore

Bad Cycle

```
class Base {
  static {
    new Child().someMethod();
  }
}

class Child extends Base {
  static final synthetic boolean $assertionsDisabled
  static { $assertionsDisabled = Child.class.get… }
  void someMethod() {
    assert 1 == 2;
  }
}

public static void main(String[] args) {
  new Child();
}
```

Bad Cycles in JaCoCo

```
class Fun {
  static final synthetic boolean[] $jacocoData = … ;

  void fun() {
    boolean[] probes = $jacocoData;
    probes[0] = true;  // NullPointerException
    …
    probes[…] = true;
  }
}
```

Bad Cycle - Solution

```
class Fun {
  static final synthetic boolean[] $jacocoData;

  static synthetic boolean[] $jacocoInit() {
    if ($jacocoData == null) $jacocoData = … ;
    return $jacocoData;
  }

  void fun() {
    boolean[] probes = $jacocoInit();
    …
```

Bad Cycle - Solution?

```
class Fun {
  static final synthetic boolean[] $jacocoData;

  static synthetic boolean[] $jacocoInit() {
    if ($jacocoData == null) $jacocoData = … ;
    return $jacocoData;
  }

  void fun() {
    boolean[] probes = $jacocoInit();
    …
```

Java Virtual Machine Specification

“6.5. **putstatic**

if the **field is final**, it must be declared in the current class, and the **instruction must occur in the <clinit>** method of the current class. **Otherwise, an IllegalAccessError** is thrown”

Checked since **JDK 9 EA b127** for class files version 53.

Bad Cycle - Correct Solution

```
class Fun {
  static ~~final~~ synthetic boolean[] $jacocoData;

  static synthetic boolean[] $jacocoInit() {
    if ($jacocoData == null) $jacocoData = … ;
    return $jacocoData;
  }

  void fun() {
    boolean[] probes = $jacocoInit();
    …
```

Interfaces

```
interface Fun {
  static final
  Object field

  static {
    field = … ;
  }
}
```

```
interface Fun {
  default method() {
    // oups???
  }
}
```

Java Language Specification

“12.4.1. When Initialization Occurs

A **class or interface** type T will be **initialized immediately before** the first occurrence of any one of the following:

…an **instance** of T is **created**…

…**static method** declared by T **is invoked**…

When a class is initialized, its superclasses are **initialized**, as well as **any superinterfaces that declare any default methods**. Initialization of an interface does not, of itself, cause initialization of any of its superinterfaces.”

Bad Cycle with Interfaces

```
interface Base {
  Object o = new Child(){}.fun();
  default void base() { } // JLS 12.4.1
}

interface Child extends Base {
  Object o = new Object() { { println(“clinit”); } };
  default Object fun() { throw new RuntimeException(); }
}

public static void main(String[] args) {
  new Child() { };  // or Child.fun();
}
```

Bad Cycle with Interfaces

- base clinit → **child method → child clinit**
- **< JDK 8u40 <=**

class: child clinit → base clinit → child method

+ **crash** because of exception

static: base clinit → **child method → child clinit**

- **< JDK 8u152 EA <=**
  child clinit → base clinit → child method

Bad Cycle with Interfaces - Solution

```
interface Fun {
  static final synthetic boolean[] $jacocoData =
    $jacocoInit();

  static synthetic boolean[] $jacocoInit() {
    return $jacocoData == null
      ? …  // slow path without assignment for JDK < 8u152
      : $jacocoData ;
  }

  default void fun() {
    boolean[] probes = $jacocoInit();
    …
```

Runtime Access

```
class RestrictiveClassLoader extends ClassLoader {
  protected Class<?> loadClass(String name, boolean resolve)
                     throws ClassNotFoundException
  {
    if (!name.startsWith("java/") &&
        !name.startsWith("org/sonarsource/"))
      throw new ClassNotFoundException();
    return super.loadClass(name, resolve);
  }
}
```

Runtime Access - Problem

```
boolean[] probes = ??? ;

// Oups
// can use only classes
// “java/**”
```

Runtime Access - Solution

```
Object access =
  java.util.UUID.$jacocoAccess;  // created at agent startup

Object[] args = new Object[] {
  classId,     // Long
  className,   // String
  probesCount  // Integer
};

access.equals(args);

boolean[] probes = (boolean[]) args[0];
```

ASM and Java 9

```
// javac -version
java version "9-ea"

// javac Fun.java

// javap -version
1.7.0_80

// javap -v Fun
major version: 53
```

```
new ClassReader(classBytes)
// IllegalArgumentException

return downgrade(classBytes)
  ? upgrade(transform(classBytes))
  : transform(classBytes);
```

Get Involved and Have Fun

- https://groups.google.com/forum/#!forum/jacoco
- StackOverflow
- https://github.com/eclipse/eclemma
- https://github.com/jacoco/jacoco


KEEP
CALM
and keep clapping
THIS
IS THE END