Java Code Coverage Mechanics

by Evgeny Mandrikov

at EclipseCon Europe 2017
Evgeny Mandrikov  Marc Hoffmann

@_Godin_  @marcandsweep
Godin  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 employers.
*
* They can only dream that they own them.
*/
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!!!
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

- runtime profiling
  - JVMPi
  - JVMTI

- instrumentation
  - source
    - offline
  - bytecode
    - on-the-fly
JaCoCo works on class files only
As easy as setting an arg for the JVM

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

class PreMain {
    public static void premain(
        String options,
        Instrumentation instrumentation
    ) throws Exception {
        instrumentation.addTransformer(...);
    }
}
```
Java Byte Code Instrumentation
JaCoCo Validation Test Suite

```java
// 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()) { // $line-missedforeachincrementer$
    nop(o); // $line-missedforeach$
}
```
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

Evgeny Mandrikov authored and committed
Not only issues in JaCoCo...

<table>
<thead>
<tr>
<th>T</th>
<th>Key</th>
<th>Summary</th>
<th>Assignee</th>
<th>Reporter</th>
<th>P ↓</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>JDK-8154017</td>
<td>Shutdown hooks are racing against shutdown sequence, if System.exit()-calling thread is interrupted</td>
<td>Chris Hegarty</td>
<td>Aleksey Shipilev</td>
<td>2</td>
<td>CLOSED</td>
</tr>
<tr>
<td></td>
<td>JDK-8164302</td>
<td>No initialization for super interface with default method</td>
<td>Karen Kinneer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>JDK-8080555</td>
<td>Different bytecode between JDK8u45 and JDK8u60-ea-b12</td>
<td>Vicente Arturo Romero Zaldivar</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>JDK-8134862</td>
<td>JVM crash at PhaseldealLoop::idom_no_update</td>
<td>Balchandra Vaidya</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>JDK-8073658</td>
<td>Invalid annotations in bridge methods</td>
<td>Srikanth Adayapalam</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>JDK-8131041</td>
<td>Garbage in output of DecimalFormat</td>
<td>Naoto Sato</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>JDK-8163969</td>
<td>Cyclic interface initialization causes JVM crash</td>
<td>Coleen Phillimore</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

```java
probes[(id) = true;
ALOADx probes
xPUSH id
ICONST_1
BASTORE
```
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();
    }
}
class Fun {
    static final synthetic boolean[] $jacocoData = ...;

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

Nice idea!
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

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

// javap -c -p -l Fun
private static synthetic lambda$fun$0() {
    ...
}
```
new ClassReader(classBytes)

// IllegallegalArgumentException

return downgrade(classBytes)

? upgrade(transform(classBytes))

: transform(classBytes);
Enjoy Hardcore
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;
    }
}
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();
        ...
    }
}
“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 IllegalAccessException is thrown”

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

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

```java
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
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();
        ...
    }
}
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)

```java
boolean[] probes = ???;
// Oups
// can use only classes
// "java/**"
```
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];
Get Involved and Have Fun

- https://jdk9.java.net/download/
- https://groups.google.com/forum/#!forum/jacoco
- StackOverflow
- https://github.com/eclipse/eclelemma
- https://github.com/jacoco/jacoco
KEEP CALM
and keep clapping
THIS IS THE END