Code Coverage in Practice

Evgeny Mandrikov, SonarSource, @_godin_
Tested on 5 major JDK versions

jacoco / jacoco

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

✓ # 1142.1 JDK=5
✓ # 1142.2 JDK=6
✓ # 1142.3 JDK=7
✓ # 1142.4 JDK=8
✓ # 1142.7 JDK=9-ea-stable
We do not only identify issues in JaCoCo...
// 5. Always executed while block
while (t()) { // $line-whiletrue$
    if (t()) {
        break;
    }
}

// 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); // $line-missedforeach$
}
EclEmma @ Eclipse

- Proposal accepted in July 2016 (https://projects.eclipse.org/projects/technology.eclemma)
- Initial contribution and IP clearance (https://github.com/eclipse/eclemma)
- **3.0.0 is part of Oxygen** (Eclipse IDE for Java developers, …)
  - almost identical functionality as 2.3.3 with new namespace org.eclipse.eclemma.*
  - updatable from 2.3.3
  - Look&Feel adjustments for better integration
  - HiDPI icons
  - still works for Eclipse 3.8 and greater
  - **Java 9** ready
What can code coverage be used for?

- White-box Testing (Unit Tests)
- Integration Testing
- Usage Analysis
- Differential Code Coverage

- JaCoCo works for any code running on the JVM
- It is not intended to be used for profiling!
Integration with Various Tools

- **IDE**
  - Eclipse EclEmma™
  - IntelliJ IDEA
  - NetBeans

- **Build**
  - Maven
  - Gradle
  - Ant

- **CI**
  - Jenkins
  - TeamCity
  - Visual Studio Team Services
  - SonarQube

- **you name it - Java API**
Demo

- Unit Testing within Eclipse IDE
- Differential Code Coverage within Eclipse IDE
Principles and Best Practices

- Don’t care about percentage value, observe the amount of untested code
- Focus on coverage of modified and new code
- Make coverage analysis an inherent part of your build/test chain
- Make coverage reports available to everybody in the team
  - But not the management!
- Always go for functional coverage when writing tests
  - Just executing code will not improve its quality!
Continuous Inspection - SonarQube

<table>
<thead>
<tr>
<th>Bugs &amp; Vulnerabilities</th>
<th>Code Smells</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 A Bugs</td>
<td>30d A Debt</td>
<td>94.0%</td>
</tr>
<tr>
<td>0 A Vulnerabilities</td>
<td>1.7k Code Smells</td>
<td>2.9k Unit Tests</td>
</tr>
<tr>
<td>0 New Bugs</td>
<td>2h New Debt</td>
<td>96.0%</td>
</tr>
<tr>
<td>0 New Vulnerabilities</td>
<td>18 New Code Smells</td>
<td>New Lines to Cover</td>
</tr>
</tbody>
</table>

Leak Period: since 4.2-SNAPSHOT
started 25 days ago
JaCoCo works on Java class files only

*.class

On-the-fly or Pre-instrumentation

*.exec

Runtime

Analysis

public static RRuntime createFor(final Instrumentation
final String className, final String accessField
throws ClassNotFoundException {
final ClassFileTransformer transformer = new ClassFileTransformer
public byte[] transform(final ClassLoader loader,
final String name, final Class<?> class
final ProtectionDomain protectionDomain
throws IllegalClassFormatException {
if (name.equals(className)) {
return instrument(source, accessFieldName);
}
return null;
}
)};
inst.addTransformer(transformer);
final Class<?> class = Class.forName(className.replace
inst.removeTransformer(transformer);
try {
try {
class.getField(accessFieldName);
} catch (final NoSuchFieldException e) {
throw new RuntimeException("Class %s could not be instrumented. ",
return new ModifiedSystemClassRuntime(class, access
}
Common Pitfalls and how to avoid them

- Different Class Files runtime/analysis
  - different compiler implementations
  - different compiler versions
  - different compiler settings
  - Pack200

- Different version of the same class in same group

- Not graceful JVM termination

- Reflection (synthetic fields and methods)

- Pre-instrumentation requires direct dependency on the JaCoCo runtime

- Interoperability with other agents
  - PowerMock might bypass agents, because reads files
  - overrides JaCoCo init method
Get Involved

- **EclEmma and JaCoCo user's group:**
  - jacoco@googlegroups.com
  - https://groups.google.com/forum/#!forum/jacoco

- **EclEmma**
  - http://www.eclemma.org/
  - https://github.com/eclipse/eclemma

- **JaCoCo**
  - https://github.com/jacoco/jacoco
Evaluate the Sessions

Sign in and vote at eclipsecon.org

-1 0 +1