Tested on 5 major JDK versions

jacoco / jacoco

Build Jobs

- #1142.1 JDK=5
- #1142.2 JDK=6
- #1142.3 JDK=7
- #1142.4 JDK=8
- #1142.7 JDK=9-ea-stable

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
We do not only identify issues in JaCoCo...
JaCoCo Validation Test Suite

// 5. Always executed while block
while (t()) { // $line-while-true$
    if (t()) {
        break;
    }
}

// 6. Executed while block
int i = 0;
while (i++ < 3) { // $line-while-true-false$
    nop(); // $line-executed-while$
}

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

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

// 9. Executed for block
for (int j = 0; j < i; j++) { // $line-executed-for-incrementer$
    nop(); // $line-executed-for$
}

// 10. Missed for each block
for (Object o : Collections.emptyList()) { // $line-missed-for-each-incrementer$
    nop(o); // $line-missed-for-each$
}
EclEmma @ Eclipse

● Proposal accepted in July 2016
  (https://projects.eclipse.org/projects/technology.eclemma)
● Initial contribution and IP clearance (https://github.com/eclipse/eclemma)
● Project infrastructure
  ○ builds
  ○ website in progress
  ○ update path in progress
  ○ etc.

Target: Become part of Oxygen!

(so you don’t have to install it any more…)
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

- Maven aggregate report
- 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

**Bugs & Vulnerabilities**
- 0 Bugs
- 0 Vulnerabilities
- 0 New Bugs
- 0 New Vulnerabilities

**Code Smells**
- 30d Debt: started 5 years ago
- 1.7k Code Smells
- 2h New Debt
- 18 New Code Smells

**Coverage**
- 94.0% Coverage
- 2.9k Unit Tests
- 96.0% Coverage
- 945 New Lines to Cover

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

On-the-fly or Pre-instrumentation

*.class

Runtime

*.exec

Analysis

```java
public static IRuntime 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)
    final String name, final Class<?> class
    final ProtectionDomain protectionDomain)
    throws IllegalArgumentException {
        if (name.equals(className)) {
            return instrument(source, accessFieldName);
        }
        return null;
    }
    };
    inst.addTransformer(transformer);
    final Class<?> clazz = Class.forName(className.replaceAll(\Q\Q\E, \Q\Q\E);
    inst.removeTransformer(transformer);
    try {
        clazz.getField(accessFieldName);
    } catch (final NoSuchFieldException e) {
        throw new RuntimeException("Class %s could not be instrumented.", className);
    }
    return new ModifiedSystemClassRuntime(clazz, accessFieldName);
}
```
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
  ○ http://www.jacoco.org/jacoco/index.html
  ○ https://github.com/jacoco/jacoco
Evaluate the Sessions
Sign in and vote at eclipsecon.org

-1 0 +1