



MUTATION TESTING

START HUNTING THE BUGS

@SVENRUPPERT

vaadin}>



SVEN RUPPERT
Developer Advocate @ Vaadin

CODING JAVA SINCE 1996
DISTRIBUTED SYSTEMS SINCE 2002
CONSULTING WORLD WIDE
JOINED VAADIN 2017



Private Sector: Automotive / Aerospace / SMB /

Public Sector: Military / Government

NonProfit / NonGov: World Bank / UN / YPARD / CGIAR

vaadin}>

The Environment

@SvenRuppert

Do you have bugs in
your code ?

no

since years
you are
working hard
on this....



Codebase is > 13 years old

no test coverage

no dedicated refactoring budget

decrease complexity

but... lets start with the basics

are you using jUnit?

assume that the following would make sense.. ;-)

```
public class Service {  
    public int add(int a, int b){  
        if(a<2){  
            return (a+b) * -1;  
        } else {  
            return a+b;  
        }  
    }  
}
```

How many tests
you will need ?

it depends ;-)

```
public class Service {  
    public int add(int a, int b){  
        if(a<2){  
            return (a+b) * -1;  
        } else {  
            return a+b;  
        }  
    }  
}
```

How many tests
you will need ?
it depends ;-)

for line 100% coverage 2

but will this be enough? **maybe ;-)**

```
public class Service {  
    public int add(int a, int b){  
        if(a<2){  
            return (a+b) * -1;  
        } else {  
            return a+b;  
        }  
    }  
}
```

for line 100% coverage 2

but will this be enough? **maybe ;-)**

How many tests
you will need ?

it depends ;-)

how to find out, what will be enough?

how to find the right tests?

```
public class Service {  
    public int add(int a, int b){  
        if(a<2){  
            return (a+b) * -1;  
        } else {  
            return a+b;  
        }  
    }  
    @Test  
    public void testAdd001() throws Exception {  
        final int add = new Service().add(0, 0);  
        Assertions.assertThat(add).isEqualTo(0);  
    }  
  
    @Test  
    public void testAdd002() throws Exception {  
        final int add = new Service().add(3, 0);  
        Assertions.assertThat(add).isEqualTo(3);  
    }  
}
```

How many tests
you will need ?

Mutation Testing is a structural testing method

we want to find a way to write "good" tests
how to find "good" tests?

let the machine find the targets

let's mutate it... but how?

a mutation is a small change in the code

.. small enough to be a small defect

P will be the program

T will be the collection of all tests / Test Suite

P will be the program

T will be the collection of all tests / Test Suite

we will create a sequence of mutations / **P1,P2,P3...**

.. **Px** will have only one mutation compared to **P**

running all tests from **T** against **Px**

green: **T** will kill the mutation

.. at least one test from **T** will fail

red: if all tests are green

Mutation Testing - the Idea

@SvenRuppert

if we kill **k** out of **n** mutants

-> we are not good enough ;-)

we are **perfect** enough if we are reaching : **k == n**

how to create all versions of Px ?

.. the good thing..

we could almost generate/
automate everything

practical TDD with Mutation Testing

generating the mutants and

running all junit tests

check the reports

write more / better tests

loop until quality target reached

mutants are a good approach / model to estimate
the default rate of defects per 1k lines of the **P**

estimate that:

the defects are independent

no need to know that Mutation Testing will be done,
independent creation of **T**

for **K==n**

we need a high number of mutants (**P1, P2, .., Px**)

.. mostly it will lead into exponential numbers of **Px**

.. how to find YOUR barrier you
have to reach?

but.. what is a mutation?

but.. what is a mutation?

Value Mutation

changing constants,
loop bounds (adding/subtracting values)

but.. what is a mutation?

Value Mutation

Decision Mutation

for example < will be changed to <=

but.. what is a mutation?

Value Mutation Decision Mutation

Statement Mutation

for example swapping/deleting/duplicating
lines of code

but.. what is a mutation?

Value Mutation Decision Mutation Statement Mutation

for **Java** you could think about more language spec. mutations

- .. changing modifiers
 - .. changing between static / non static
- .. delete member initialization
 - .. delete this.
- .. argument order change

mutation testing is an add on to normal jUnit TDD

tools are supporting it well

generating and running all tests are time consuming

but most important

will effect your project structure

muJava

- 
- 2003.** First released as JMutation (Java Mutation System).
 - 2004.** The name was changed to MuJava (Mutation System).
 - 2005.** Software Copyright Registration, ALL RIGHTS RESERVED.
 - 2005.** Version 2 released with several fault fixes and new mutation operators.
 - 2008.** Version 3 released with minimal support for Java 1.5 and 1.6.
 - 2013.** Version 4 released to support JUnit tests and Java 1.6 language features, including generics, annotations, enumerations, varargs, enhanced for-each loops, and static imports.
 - 2015.** Additional and improved error messages. Bug fixes for OpenJava. Licensing changed to the Apache license.

<https://cs.gmu.edu/~offutt/mujava/>

<https://github.com/jeffoffutt/muJava/graphs/contributors>

<http://pitest.org/>

2012. started around 2012 with a small codebase.

2014. very active since 2014

active ;-)

A yellow rectangular bar is positioned diagonally across the slide. It contains the text "active ;-)" in a white, sans-serif font. The bar has a slight texture or noise effect applied to its surface.

<http://pitest.org/>

assume the following would make sense ;-)

```
public class Service {  
    public int add(int a, int b){  
        if (a<2) {  
            return (a+b) * -1;  
        } else {  
            return a+b;  
        }  
    }  
}
```

Mutation Testing - Hello World

@SvenRuppert

```
public class Service {  
    public int add(int a, int b){  
        if (a<2) {  
            return (a+b) * -1;  
        } else {  
            return a+b;  
        }  
    }  
}
```

how many test you will need for..

100% Line Coverage... and...

to be save ?

2 for Line Coverage

we will see ;-)

Mutation Testing - Hello World

@SvenRuppert

```
public class Service {  
    public int add(int a, int b){  
        if (a<2) {  
            return (a+b) * -1;  
        } else {  
            return a+b;  
        }  
    }  
}
```

100% Line Coverage... and...

we have one if statement with an else branch

this will lead to 2 jUnit Tests to get 100 %

Mutation Testing - Hello World

@SvenRuppert

```
public class Service {  
    public int add(int a, int b){  
        if (a<2) { return (a+b) * -1; }  
        else    { return a+b;      }  
    }  
}
```

100% Line Coverage... and...

@Test

```
public void testAdd001() throws Exception {  
    final int add = new Service().add(0, 0);  
    Assertions.assertThat(add).isEqualTo(0);  
}
```

@Test

```
public void testAdd002() throws Exception {  
    final int add = new Service().add(3, 0);  
    Assertions.assertThat(add).isEqualTo(3);  
}
```

Mutation Testing - Hello World

@SvenRuppert

```
final int add = new Service().add(0, 0);
Assertions.assertThat(add).isEqualTo(0);
```

```
6. public class Service {
7.     public int add(int a, int b) {
8.         if (a < 2) {
9.             return (a + b) * -1;
10.        } else {
11.            return a + b;
12.        }
13.    }
```

```
final int add = new Service().add(3, 0);
Assertions.assertThat(add).isEqualTo(3);
```

```
6. public class Service {
7.     public int add(int a, int b) {
8.         if (a < 2) {
9.             return (a + b) * -1;
10.        } else {
11.            return a + b;
12.        }
13.    }
```

Mutation Testing - Hello World

@SvenRuppert

```
final int add = new Service().add(0, 0);
Assertions.assertThat(add).isEqualTo(0);
final int add = new Service().add(3, 0);
Assertions.assertThat(add).isEqualTo(3);
```

we got 100% Line Coverage

How good these tests are?

How to measure if these test are good?

How to find the good tests?

```
final int add = new Service().add(0, 0);
```

```
final int add = new Service().add(3, 0);
```

How to find the good tests?

let's generate a the mutation report

with maven : **pitest: mutationCoverage**

>> Generated 54 mutations Killed 3

```
org.pitest.....mutators.ConditionalsBoundaryMutator  
org.pitest.....mutators.IncrementsMutator  
org.pitest.....mutators.ReturnValsMutator  
org.pitest.....mutators.MathMutator  
org.pitest.....mutators.NegateConditionalsMutator
```

Mutation Testing - Hello World

@SvenRuppert

```
final int add = new Service().add(0, 0);
final int add = new Service().add(3, 0);
```

>> Generated 54 mutations Killed 3

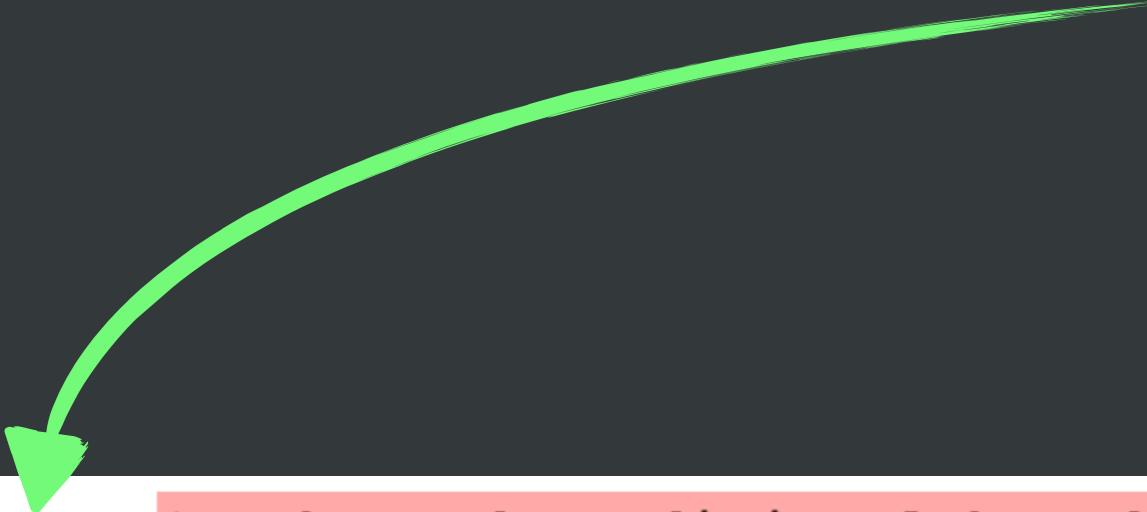
Breakdown	Name	Coverage
	Service.java	3/7
6	public class Service {	
7	public int add(int a, int b) {	
8	if (a < 2) {	
9	return (a + b) * -1;	3/7
10	} else {	
11	return a + b;	
12	}	
13	}	

Mutation Testing - Hello World

@SvenRuppert

```
final int add = new Service().add(0, 0);  
final int add = new Service().add(3, 0);
```

>> Generated 54 mutations Killed 3



```
8 2 if (a < 2) {  
9 3     return (a + b) * -1;  
10 } else {  
11 2     return a + b;
```

8

1. changed conditional boundary → SURVIVED
2. negated conditional → KILLED

9

1. Replaced integer addition with subtraction → SURVIVED
2. Replaced integer multiplication with division → SURVIVED
3. replaced return of integer sized value with `(x == 0 ? 1 : 0)` → KILLED

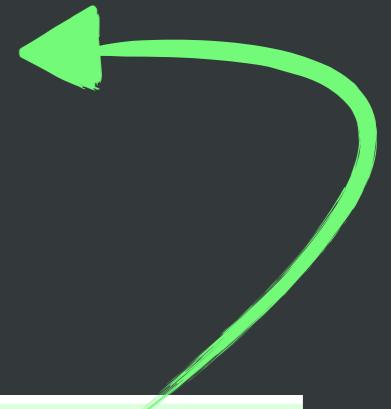
11

1. Replaced integer addition with subtraction → SURVIVED
2. replaced return of integer sized value with `(x == 0 ? 1 : 0)` → KILLED

Mutation Testing - Hello World

@SvenRuppert

```
final int add = new Service().add(0, 0);  
final int add = new Service().add(3, 0);  
final int add = new Service().add(2, 0);
```



>> Generated 54 mutations

Killed 4

```
8 2 if (a < 2) {  
9 3     return (a + b) * -1;  
10 } else {  
11 2     return a + b;  
12 }
```

8 1. changed conditional boundary → KILLED

2. negated conditional → KILLED

9 1. Replaced integer addition with subtraction → SURVIVED

2. Replaced integer multiplication with division → SURVIVED

3. replaced return of integer sized value with (x == 0 ? 1 : 0) → KILLED

11 1. Replaced integer addition with subtraction → SURVIVED

2. replaced return of integer sized value with (x == 0 ? 1 : 0) → KILLED

Mutation Testing - Hello World

@SvenRuppert

```
final int add = new Service().add(2, 0);
final int add = new Service().add(1, 1);
```

>> Generated 54 mutations Killed 5

killed 9:1

```
8 2 if (a < 2) {
9 3     return (a + b) * -1;
10 } else {
11 2     return a + b;
12 }
```

8 1. changed conditional boundary → KILLED

2. negated conditional → KILLED

9 1. Replaced integer addition with subtraction → KILLED

2. Replaced integer multiplication with division → SURVIVED

3. replaced return of integer sized value with (x == 0 ? 1 : 0) → KILLED

11 1. Replaced integer addition with subtraction → SURVIVED

2. replaced return of integer sized value with (x == 0 ? 1 : 0) → KILLED

Mutation Testing - Hello World

@SvenRuppert

```
final int add = new Service().add(2, 0);
final int add = new Service().add(1, 1);
final int add = new Service().add(2, 2);
```

>> Generated 54 mutations Killed 6

killed 9:11:1

```
8 2 if (a < 2) {
9 3     return (a + b) * -1;
10 } else {
11 2     return a + b;
12 }
```

8 1. changed conditional boundary → KILLED

2. negated conditional → KILLED

9 1. Replaced integer addition with subtraction → KILLED

2. Replaced integer multiplication with division → SURVIVED

3. replaced return of integer sized value with ($x == 0 ? 1 : 0$) → KILLED

11 1. Replaced integer addition with subtraction → KILLED

2. replaced return of integer sized value with ($x == 0 ? 1 : 0$) → KILLED

Mutation Testing - Hello World

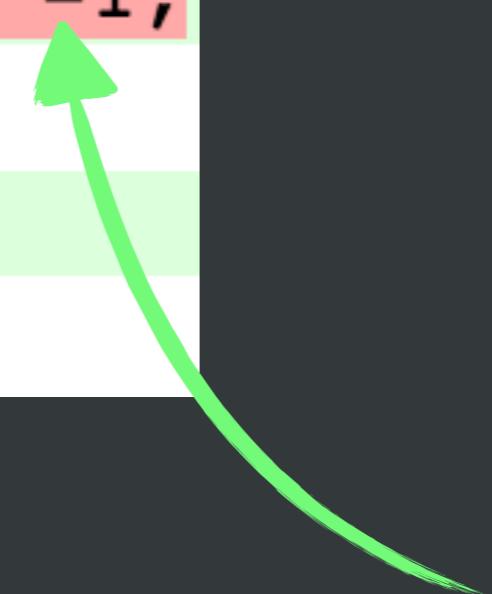
@SvenRuppert

```
final int add = new Service().add(1, 1);
final int add = new Service().add(2, 2);
```

>> Generated 54 mutations Killed 6

8	1. changed conditional boundary → KILLED
	2. negated conditional → KILLED
9	1. Replaced integer addition with subtraction → KILLED
	2. Replaced integer multiplication with division → SURVIVED
	3. replaced return of integer sized value with (x == 0 ? 1 : 0) → KILLED
11	1. Replaced integer addition with subtraction → KILLED
	2. replaced return of integer sized value with (x == 0 ? 1 : 0) → KILLED

```
8 2 if (a < 2) {
9 3     return (a + b) * -1;
10 } else {
11 2     return a + b;
12 }
```



mutation tests are often leading to

...cleaner code compared to jUnit only

Mutation Testing - Lesson Learned

@SvenRuppert

```
public static final String[] discardCommonPrefix(String a, String b) {  
    String[] ret = { a, b };  
    int l = a.length() < b.length() ? a.length() : b.length();  
    for (int i = 0; i < l; i++) {  
        if (a.charAt(i) == b.charAt(i)) {  
            if (i + 1 < l) { ret[0] = a.substring(i + 1); ret[1] = b.substring(i + 1); }  
            else {  
                if (a.length() < b.length()) { ret[0] = ""; ret[1] = b.substring(i + 1); }  
                if (a.length() == b.length()) { ret[0] = ""; ret[1] = ","; }  
                if (a.length() > b.length()) { ret[0] = a.substring(i + 1); ret[1] = ","; }  
            }  
        } else break;  
    }  
    return ret;  
}
```

Mutation Testing - Lesson Learned

@SvenRuppert

```
public String[] discardCommonPrefix(String a, String b) {  
    final String[] ret = new String[2];  
    int l;  
    if (a.length() < b.length()) { l = a.length(); }  
    else  
        { l = b.length(); }  
  
    int position = 0;  
    for (; position < l; position++) {  
        final char charA = a.charAt(position);  
        final char charB = b.charAt(position);  
        if (charA != charB) { break; }  
    }  
  
    if (position >= a.length()) { ret[0] = ""; }  
    else  
        { ret[0] = a.substring(position); }  
  
    if (position >= b.length()) { ret[1] = ""; }  
    else  
        { ret[1] = b.substring(position); }  
    return ret;  
}
```

Mutation Testing - Lesson Learned

@SvenRuppert

Version 1

```
for {  
    if {  
        if  
        else {  
            if  
            if  
            if  
        }  
    } else  
}
```

Version 2

```
if else  
  
for {  
    if  
}  
  
if else  
if else
```

mutation tests are often leading to

- ...cleaner code compared to jUnit only
- ... smaller modules (shorter mutation runtime)

and something nice...

helps to find useless code

Example of useless Code

@SvenRuppert

```
12 public class ReflectionUtils extends org.reflections.ReflectionUtils {  
13  
14     public boolean checkInterface(final Type aClass, Class targetInterface) {  
15     2     if (aClass.equals(targetInterface)) return true;  
16  
17     final Type[] genericInterfaces = ((Class) aClass).getGenericInterfaces();  
18 2     if (genericInterfaces.length > 0) {  
19 3     for (Type genericInterface : genericInterfaces) {  
20  
22         final Type[] nextLevBackArray = ((Class) genericInterface).getGenericInterfaces();  
23 2         if (nextLevBackArray.length > 0)  
24 3         for (Type type : nextLevBackArray) {  
25 2             if (checkInterface(type, targetInterface)) return true;  
26         }  
27     }  
28 }  
29     final Type genericSuperclass = ((Class) aClass).getGenericSuperclass();  
30 1     if (genericSuperclass != null) {  
31 2     if (checkInterface(genericSuperclass, targetInterface)) return true;  
32     }  
33  
34  
35 1     return false;  
36 }
```

you need jUnit - to generate the reference

add the pitest-plugin to the build section

configure the plugin

generate the reference -> clean , install

run **pitest: mutationCoverage**

report will be under **target/pit-reports**

Mutation Testing - How to start

@SvenRuppert

pom.xml - example - build

```
<plugin>
  <groupId>org.pitest</groupId>
  <artifactId>pitest-maven</artifactId>
  <configuration>
    <outputFormats>
      <outputFormat>XML</outputFormat>
      <outputFormat>HTML</outputFormat>
    </outputFormats>
    <targetClasses>
      <param>org.rapidpm.*</param>
    </targetClasses>
    <targetTests>
      <param>org.rapidpm.*</param>
      <param>junit.org.rapidpm.*</param>
    </targetTests>
  </configuration>
</plugin>
```

Mutation Testing - How to start

@SvenRuppert

pom.xml - example - reporting

```
<reporting>
  <plugins>
    <plugin>
      <groupId>org.pitest</groupId>
      <artifactId>pitest-maven</artifactId>
      <reportSets>
        <reportSet>
          <reports>
            <report>report</report>
          </reports>
        </reportSet>
      </reportSets>
    </plugin>
  </plugins>
</reporting>
```

Start with some tests
generate the pitest report
write more tests to kill mutations
if you have time, eliminate useless tests
do it one by one

Mutation 001

Survived> **Killed**

Mutation 002

Survived> **Killed**

Mutation 003

Survived> **Killed**

Mutation 004

Survived> **Killed**

Mutation Testing - Real Code Examples

@SvenRuppert

```
Tests run: 86, Failures: 0, Errors: 0, Skipped: 0
```

```
[INFO] -----  
[INFO] Reactor Summary:  
[INFO]  
[INFO] RapidPM Dynamic Dependency Injection ..... SUCCESS [ 3.117 s]  
[INFO] rapidpm-dynamic-cdi-bom ..... SUCCESS [ 0.623 s]  
[INFO] rapidpm-dynamic-cdi-modules ..... SUCCESS [ 0.557 s]  
[INFO] rapidpm-dynamic-cdi-modules-core ..... SUCCESS [ 39.017 s]  
[INFO] -----  
[INFO] BUILD SUCCESS  
[INFO]  
[INFO] Total time: 43.526 s  
[INFO] Finished at: 2018-03-01T12:09:33+01:00  
[INFO] Final Memory: 29M/656M  
[INFO] -----
```

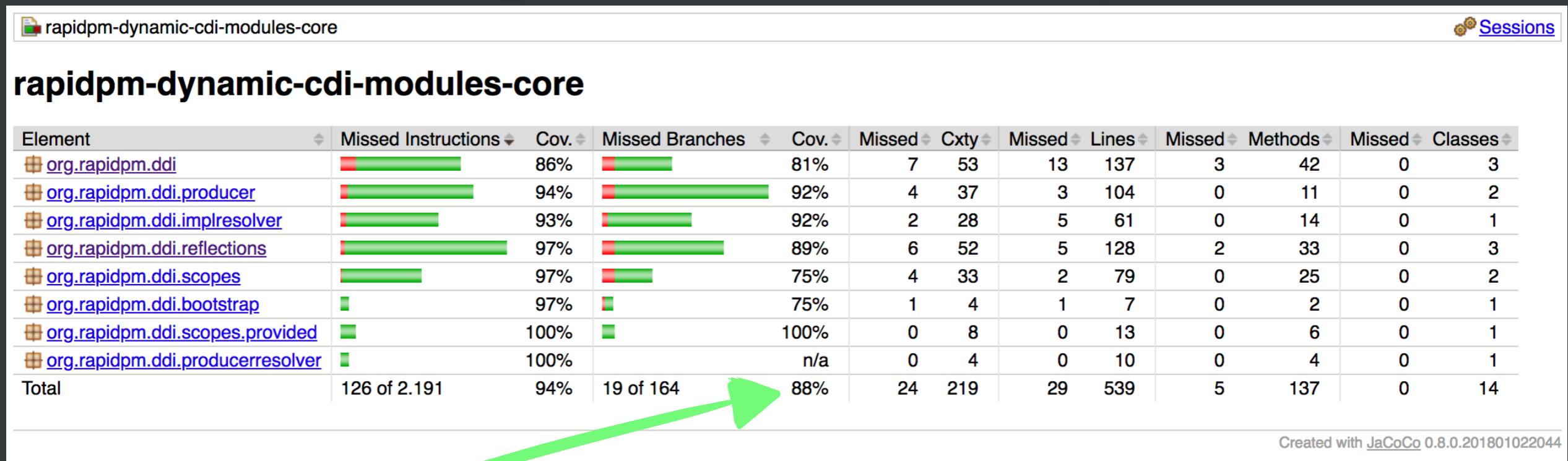


```
[INFO] -----  
[INFO] Reactor Summary:  
[INFO]  
[INFO] RapidPM Dynamic Dependency Injection ..... SUCCESS [ 1.034 s]  
[INFO] rapidpm-dynamic-cdi-bom ..... SUCCESS [ 0.021 s]  
[INFO] rapidpm-dynamic-cdi-modules ..... SUCCESS [ 0.019 s]  
[INFO] rapidpm-dynamic-cdi-modules-core ..... SUCCESS [03:36 min]  
[INFO] -----  
[INFO] BUILD SUCCESS  
[INFO]  
[INFO] Total time: 03:37 min  
[INFO] Finished at: 2018-03-01T11:56:30+01:00  
[INFO] Final Memory: 15M/326M  
[INFO] -----
```



Mutation Testing - Real Code Examples

@SvenRuppert



Pit Test Coverage Report

Project Summary

Number of Classes	Line Coverage	Mutation Coverage
11	94%	84%

Breakdown by Package

Name	Number of Classes	Line Coverage	Mutation Coverage
org.rapidpm.ddi	1	92%	84%
org.rapidpm.ddi.bootstrap	1	86%	100%
org.rapidpm.ddi.implresolver	1	92%	81%
org.rapidpm.ddi.producer	2	95%	80%
org.rapidpm.ddi.producerresolver	1	100%	100%
org.rapidpm.ddi.reflections	3	96%	89%
org.rapidpm.ddi.scopes	1	95%	78%
org.rapidpm.ddi.scopes.provided	1	100%	100%

Report generated by PIT 1.3.2

JAVA ONLY ?

ONE TOOL .. DIFFERENT LANGUAGES

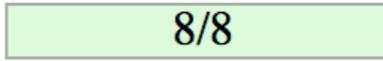
```
public class MainJava {  
  
    public int add(int a, int b) {  
        if (a < 2) {  
            return a + b * -1;  
        } else {  
            return a + b;  
        }  
    }  
}
```

```
class MainKotlin {  
    fun add(a: Int, b: Int): Int {  
        return if (a < 2) {  
            a + b * -1  
        } else {  
            a + b  
        }  
    }  
}
```

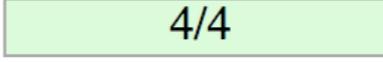
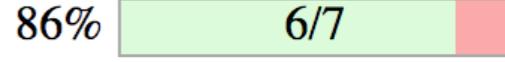
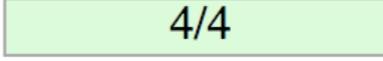
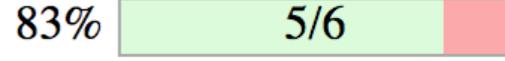
Pit Test Coverage Report

Package Summary

org.rapidpm.kotlin

Number of Classes	Line Coverage	Mutation Coverage
2	100%  8/8	85%  11/13

Breakdown by Class

Name	Line Coverage	Mutation Coverage
MainJava.java	100%  4/4	86%  6/7
MainKotlin.kt	100%  4/4	83%  5/6

Report generated by [PIT](#) 1.4.0

```
public class MainJava {  
  
    public int add(int a, int b) {  
        if (a < 2) {  
            return a + b * -1;  
        } else {  
            return a + b;  
        }  
    }  
}
```

```
class MainKotlin {  
  
    fun add(a: Int, b: Int): Int {  
        return if (a < 2) {  
            a + b * -1  
        } else {  
            a + b  
        }  
    }  
}
```

MainJava.java

```
1 package org.rapidpm.kotlin;  
2  
3 public class MainJava {  
4  
5     public int add(int a, int b) {  
6         2 if (a < 2) {  
7             3 return a + b * -1;  
8         } else {  
9             2 return a + b;  
10     }  
11 }  
12 }
```

Mutations

- 6
 - 1. changed conditional boundary → KILLED
 - 2. negated conditional → KILLED
- 7
 - 1. Replaced integer multiplication with division → SURVIVED
 - 2. Replaced integer addition with subtraction → KILLED
 - 3. replaced return of integer sized value with `(x == 0 ? 1 : 0)` → KILLED
- 9
 - 1. Replaced integer addition with subtraction → KILLED
 - 2. replaced return of integer sized value with `(x == 0 ? 1 : 0)` → KILLED

```
public class MainJava {  
  
    public int add(int a, int b) {  
        if (a < 2) {  
            return a + b * -1;  
        } else {  
            return a + b;  
        }  
    }  
  
}  
  
class MainKotlin {  
    fun add(a: Int, b: Int): Int {  
        return if (a < 2) {  
            a + b * -1  
        } else {  
            a + b  
        }  
    }  
}
```

MainKotlin.kt

```
1 package org.rapidpm.kotlin  
2  
3 class MainKotlin {  
4     fun add(a: Int, b: Int): Int {  
5         3 return if (a < 2) {  
6             2 a + b * -1  
7         } else {  
8             1 a + b  
9         }  
10    }  
11 }
```

Mutations

- 1. changed conditional boundary → KILLED
- 5. negated conditional → KILLED
- 3. replaced return of integer sized value with `(x == 0 ? 1 : 0)` → KILLED
- 6. 1. Replaced integer multiplication with division → SURVIVED
- 2. Replaced integer addition with subtraction → KILLED
- 8. 1. Replaced integer addition with subtraction → KILLED

Summary

@SvenRuppert

please, follow me ;-)

Thank You !!!

@SvenRuppert