Mutation Testing

Michael Inden

Agenda

- Mutation Testing at a glance
- Install and use PiTest
- More Info

Mutation Testing at a glance

Mutation Testing Worth Knowing

What is Mutation Testing? Why is it helpful?

- We have learned about JUnit 5 and AssertJ for writing understandable, maintainable tests.
- In addition, we can analyze code coverage with various tools
- What is missing?
- How do we actually know that the tests also detect (all) errors?

Mutation Testing Worth Knowing

We want to see if the tests are verifying the right thing

- To do this, the source code is mutated at certain points and it is checked whether the tests detect these changes.
- If the tests **detect** the **change**, then the mutation is said to be **killed**, otherwise the mutation is **live** and the tests should be corrected or supplemented.
- Limits: replace for example < by <=

Original	Mutation
<	<=
<=	<
>	>=
>=	>

Mutation Testing Worth Knowing

- Return values are mutated according to the table, for example false -> true, true -> false

Original	Mutation
Boolean value	!value
Int, short, byte value	0 -> 1, all other values -> 0
Long value	value + 1
Float, double	value + 1.0, special case NaN -> 0

PiTest

PiTest Worth Knowing

PiTest has some advantages over other tools:

- Free download at https://pitest.org/
- It is actively under development
- Many times faster compared to previous generations of tools
- Thus finally suitable for practical use

- Easy integration into build tools and development environments
- Existence of an actively developed SonarQube plug-in
- Extensive configuration possibilities
- Supports incremental analysis

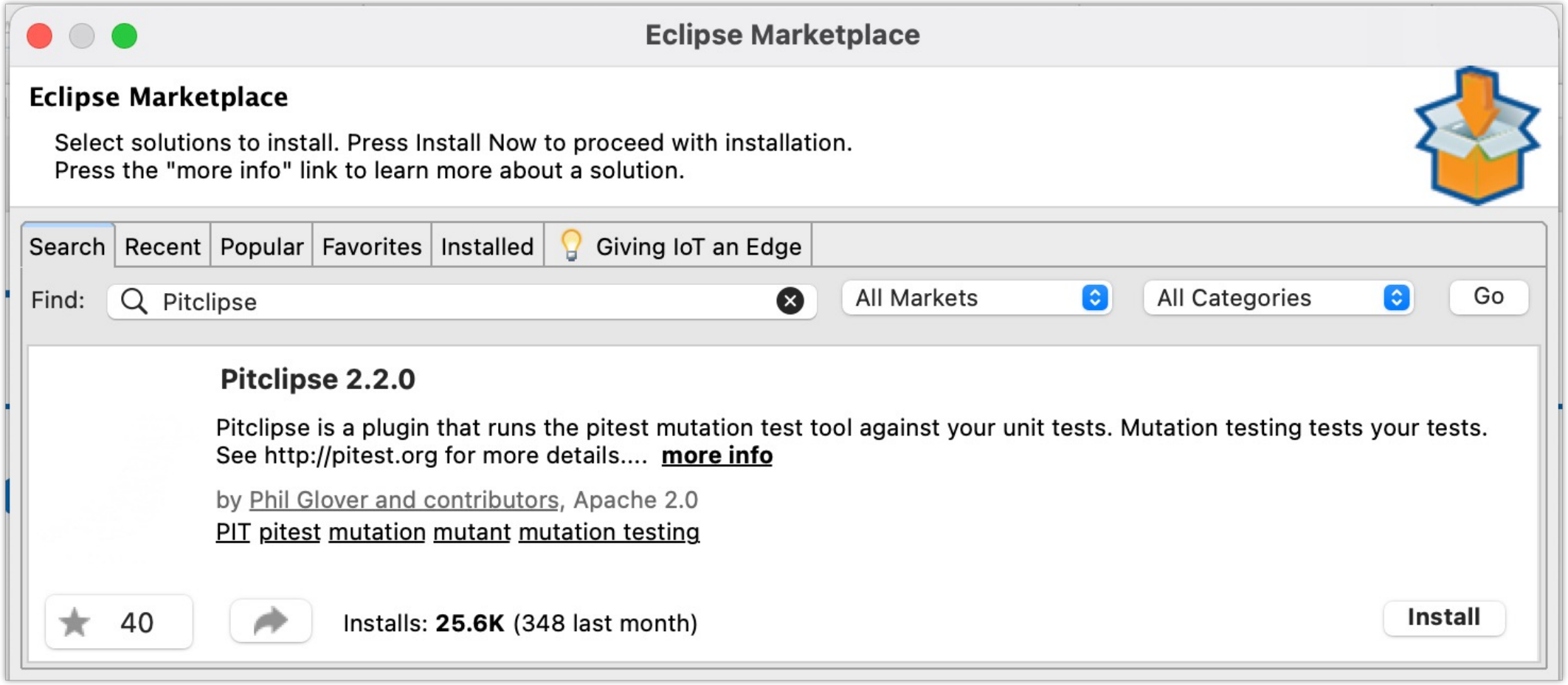
PiTest Maven Dependencies

```
<plugin>
	<groupId>org.pitest</groupId>
	<artifactId>pitest-maven</artifactId>
	<version>1.10.3</version>
	<dependencies>
		<dependency>
			<groupId>org.pitest</groupId>
			<artifactId>pitest-junit5-plugin</artifactId>
			<version>1.1.0</version>
		</dependency>
	</dependencies>
</plugin>
```

```
mvn org.pitest:pitest-maven:mutationCoverage
```

PiTest Eclipse Plugin

Installation via Eclipse Marketplace



Example

```
class SpecialCounter
{
    private int count;

    public void countIfHundredOrAbove(final int value)
    {
        if (value >= 100)
        {
            count++;
        }
    }

    public void reset()
    {
        count = 0;
    }

    public int currentCount()
    {
        return count;
    }
}
```

Idea based on https://www.oracle.com/corporate/features/mutation-testing.html

Problem «Code Coverage 100 %»

```
public class SpecialCounterTest
{
    // VERY BAD TEST ... 100% Coverage, but NO semantical check
    @Test
    @DisplayName("Boss says he wants 100% coverage. Here you go!")
    public void veryBadTrickyAssertNothing()
    {
        SpecialCounter counter = new SpecialCounter();

        counter.reset();
        counter.countIfHundredOrAbove(111);
        counter.countIfHundredOrAbove(99);

        counter.currentCount();
    }
}
```

"WE CAN'T RELY ON CODE COVERAGE"

IT DOES NOT SHOW US THE CODE THAT WAS TESTED, BUT ONLY THE CODE THAT WAS RUN!

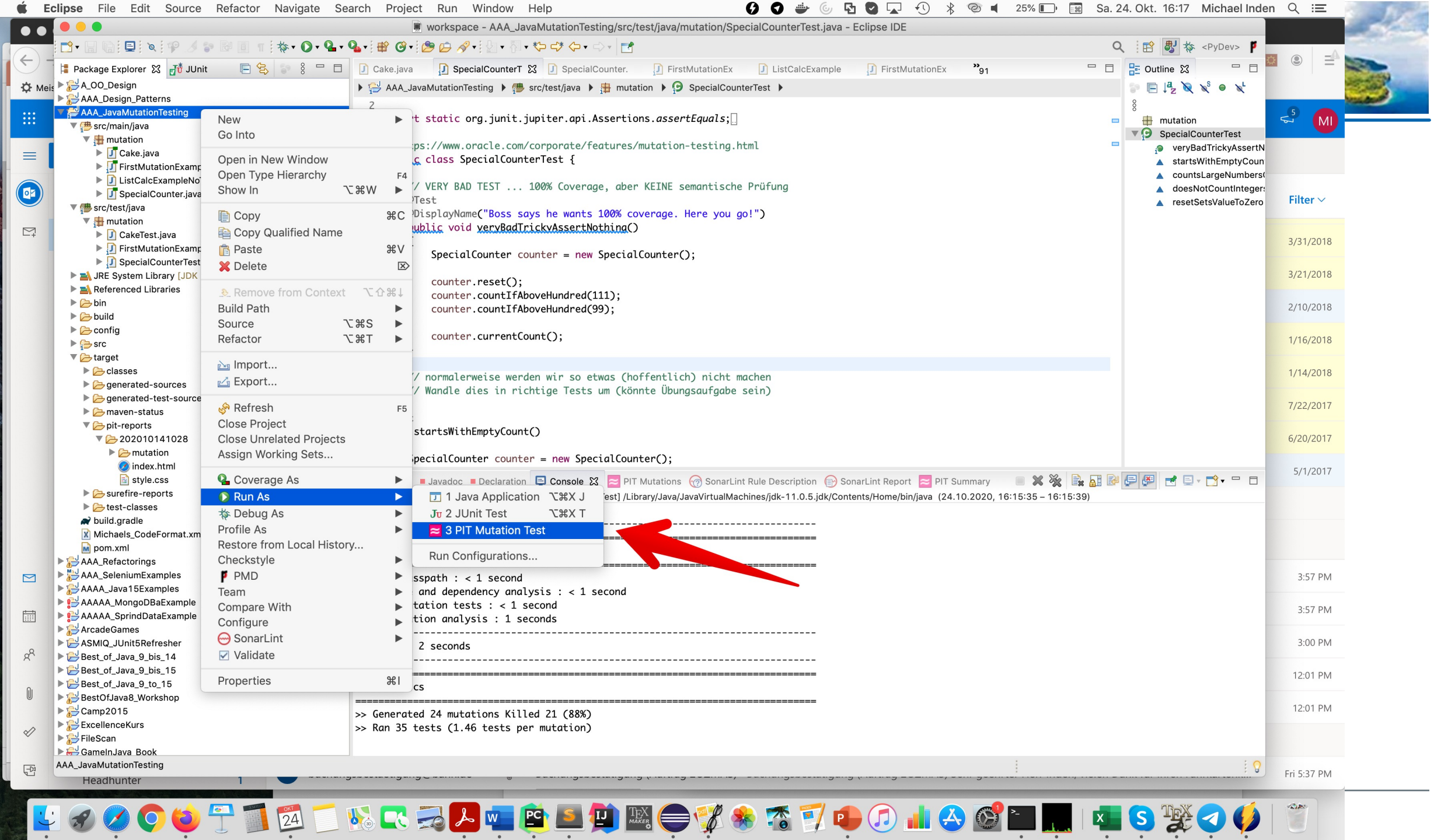
Idea based on https://www.oracle.com/corporate/features/mutation-testing.html

Remedy Meaningful Test Cases

```
@Test
void startsWithEmptyCount()
{
    SpecialCounter counter = new SpecialCounter();

    assertEquals(0, counter.currentCount());
}

@Test
void countsLargeNumbersCorrectly()
{
    SpecialCounter counter = new SpecialCounter();

    counter.countIfHundredOrAbove(111);
    counter.countIfHundredOrAbove(333);

    assertEquals(2, counter.currentCount());
}
```

Remedy Meaningful Test Cases

```
@Test
void doesNotCountIntegersBelowHundred()
{
    SpecialCounter counter = new SpecialCounter();

    counter.countIfHundredOrAbove(77);

    assertEquals(0, counter.currentCount());
}

@Test
void resetSetsValueToZero()
{
    SpecialCounter counter = new SpecialCounter();

    counter.countIfHundredOrAbove(420);
    counter.reset();

    assertEquals(0, counter.currentCount());
}
```


Eclipse File Edit Source Refactor Navigate Search Project Run Window Help
Sa. 24. Okt. 16:17 Michael Inden
workspace - AAA_JavaMutationTesting/src/test/java/mutation/SpecialCounterTest.java - Eclipse IDE
Package Explorer
JUnit
A_OO_Design
AAA_Design_Patterns
AAA_JavaMutationTesting
src/main/java
mutation
Cake.java
FirstMutationExamp
ListCalcExampleNo
SpecialCounter.java
src/test/java
mutation
CakeTest.java
FirstMutationExamp
SpecialCounterTest
JRE System Library [JDK
Referenced Libraries
bin
build
config
src
target
classes
generated-sources
generated-test-source
maven-status
pit-reports
202010141028
mutation
index.html
style.css
surefire-reports
test-classes
build.gradle
Michaels_CodeFormat.xm
pom.xml
AAA_Refactorings
AAA_SeleniumExamples
AAAA_Java15Examples
AAAAA_MongoDBaExample
AAAAA_SprindDataExample
ArcadeGames
ASMIQ_JUnit5Refresher
Best_of_Java_9_bis_14
Best_of_Java_9_bis_15
Best_of_Java_9_to_15
BestOfJava8_Workshop
Camp2015
ExcellenceKurs
FileScan
GameInJava_Book
AAA_JavaMutationTesting
New
Go Into
Open in New Window
Open Type Hierarchy F4
Show In ⌥⌘W
Copy ⌘C
Copy Qualified Name
Paste ⌘V
Delete
Remove from Context ⌥⇧⌘↓
Build Path
Source ⌥⌘S
Refactor ⌥⌘T
Import...
Export...
Refresh F5
Close Project
Close Unrelated Projects
Assign Working Sets...
Coverage As
Run As
Debug As
Profile As
Restore from Local History...
Checkstyle
PMD
Team
Compare With
Configure
SonarLint
Validate
Properties ⌘I
1 Java Application ⌥⌘X J
2 JUnit Test ⌥⌘X T
3 PIT Mutation Test
Run Configurations...
Cake.java
SpecialCounterT
SpecialCounter.
FirstMutationEx
ListCalcExample
FirstMutationEx
AAA_JavaMutationTesting
src/test/java
mutation
SpecialCounterTest
Outline
mutation
SpecialCounterTest
veryBadTrickyAssertN
startsWithEmptyCoun
countsLargeNumbers(
doesNotCountIntegers
resetSetsValueToZero
Javadoc
Declaration
Console
PIT Mutations
SonarLint Rule Description
SonarLint Report
PIT Summary
>> Generated 24 mutations Killed 21 (88%)
>> Ran 35 tests (1.46 tests per mutation)

Mutation Testing in Action

Problems | Javadoc | Declaration | Console | PIT Mutations | SonarLint Rule Description | SonarLint Report | PIT Summary

Pit Test Coverage Report

Project Summary

Number of Classes	Line Coverage		Mutation Coverage	
3	98%	41/42	88%	21/24

Breakdown by Package

Name	Number of Classes	Line Coverage		Mutation Coverage	
mutation	3	98%	41/42	88%	21/24

Report generated by PIT 1.4.11

Mutation Testing in Action

Problems ■ Javadoc ■ Declaration Console PIT Mutations SonarLint Rule Description SonarLint Report PIT Summary

Pit Test Coverage Report

Package Summary

mutation

Number of Classes	Line Coverage		Mutation Coverage	
3	98%	41/42	88%	21/24

Breakdown by Class

Name	Line Coverage		Mutation Coverage	
Cake.java	100%	31/31	94%	15/16
FirstMutationExample.java	75%	3/4	75%	3/4
SpecialCounter.java	100%	7/7	75%	3/4

Report generated by PIT 1.4.11

```
4   class SpecialCounter
5   {
6           private int count;
7
8           public void countIfHundredOrAbove(final int value)
9           {
10  2               if (value >= 100)
11                  {
12  1                       count++;
13                  }
14          }
15
16          public void reset()
17          {
18                  count = 0;
19          }
20
21          public int currentCount()
22          {
23  1               return count;
24          }
25  }
```

Mutations

10
1. changed conditional boundary → SURVIVED
2. negated conditional → KILLED

12
1. Replaced integer addition with subtraction → KILLED

23
1. replaced int return with 0 for mutation/SpecialCounter::currentCount → KILLED

Mutation Testing in Action

```
// MUTATION TESTING FINDS THE MISSING TEST FOR THE BOUNDARY

@Test
void countsIntegersCorrectlyForMinimumOf100()
{
    SpecialCounter counter = new SpecialCounter();
    counter.countIfHundredOrAbove(100);

    assertEquals(1, counter.currentCount());
}
```

```
4   class SpecialCounter
5   {
6           private int count;
7
8           public void countIfHundredOrAbove(final int value)
9           {
10  2               if (value >= 100)
11                  {
12  1                       count++;
13                  }
14          }
15
16          public void reset()
17          {
18                  count = 0;
19          }
20
21          public int currentCount()
22          {
23  1               return count;
24          }
25  }
```

Mutations

10	1. changed conditional boundary → KILLED 2. negated conditional → KILLED
12	1. Replaced integer addition with subtraction → KILLED
23	1. replaced int return with 0 for mutation/SpecialCounter::currentCount → KILLED

Further Info

- https://pitest.org/quickstart/
- https://www.codeaffine.com/2015/10/05/what-the-heck-is-mutation-testing/
- https://jaxenter.de/mutant-testing-pit-java-84437
- https://www.baeldung.com/java-mutation-testing-with-pitest
- https://blog.scottlogic.com/2017/09/25/mutation-testing.html
- https://www.oracle.com/corporate/features/mutation-testing.html
- https://dzone.com/articles/mutation-testing-covering-your-code-with-right-tes
- https://dzone.com/articles/mutation-testing-covering-your-code-with-right-tes-1

Thank You