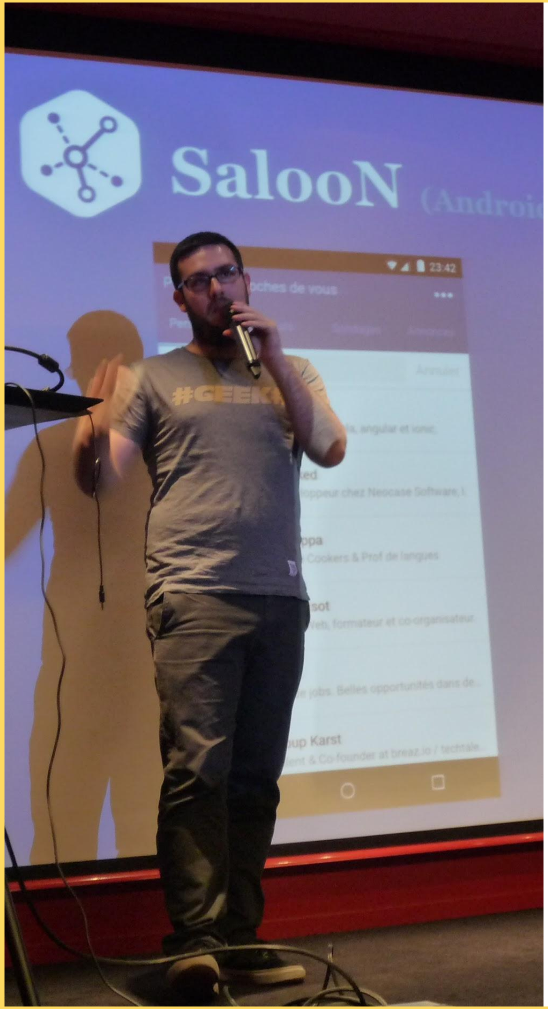
Mutation testing

Gotta Kill 'Em All !

Loïc Knuchel

RIVIERADEV



Loïc Knuchel

@loicknuchel

Développeur Scala chez criteo.

Organisateur des Human Talks Paris

Software craftsman

Hexagonal architecture

FP

CQRS

Event Storming

DDD

Property based testing

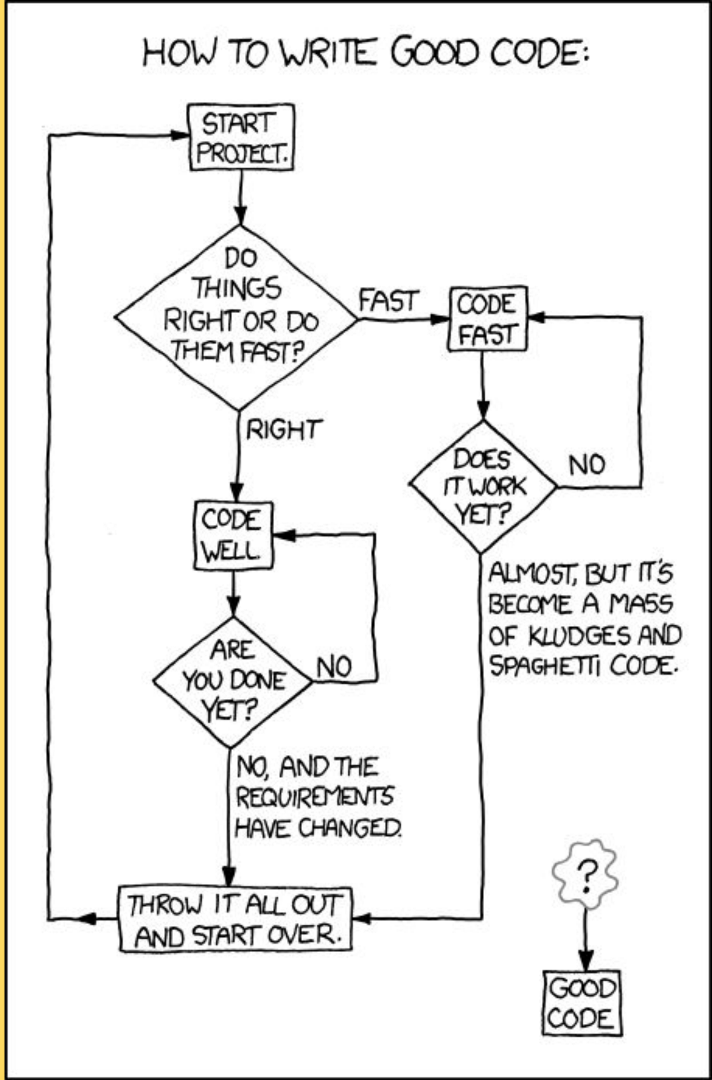
loicknuchel@gmail.com

Living documentation

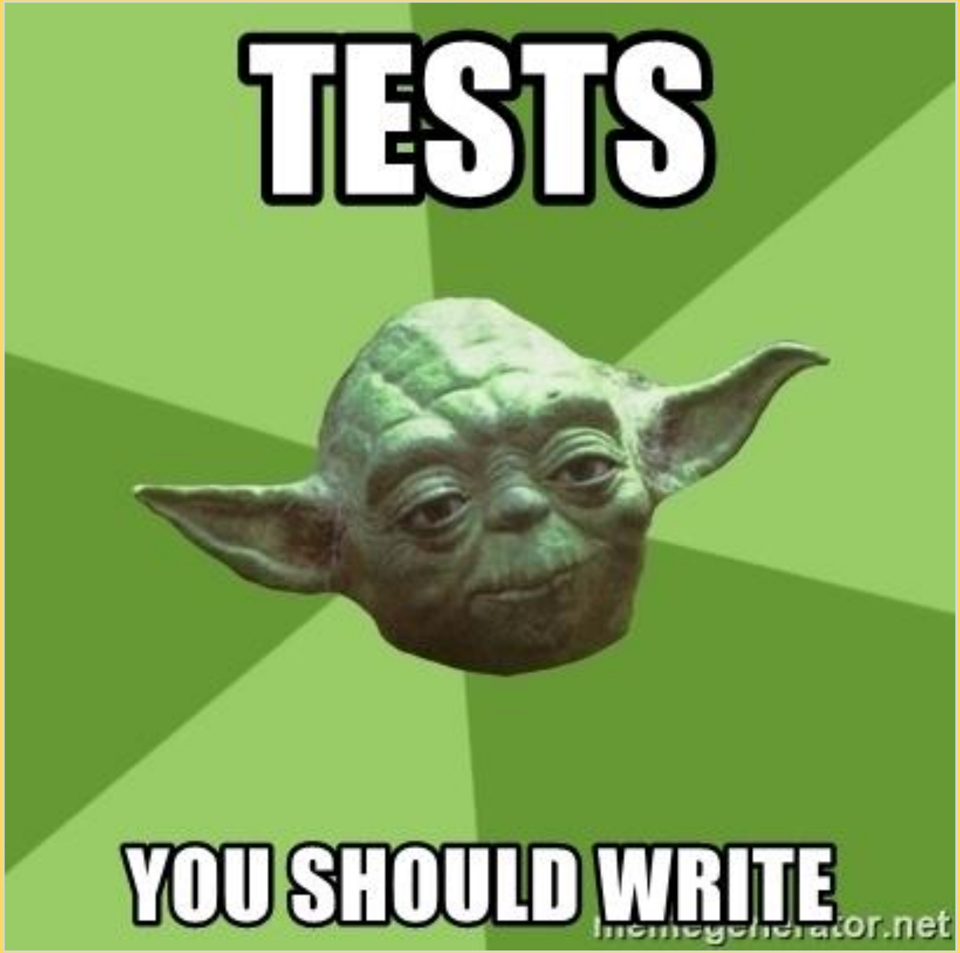
Event Sourcing

TDD

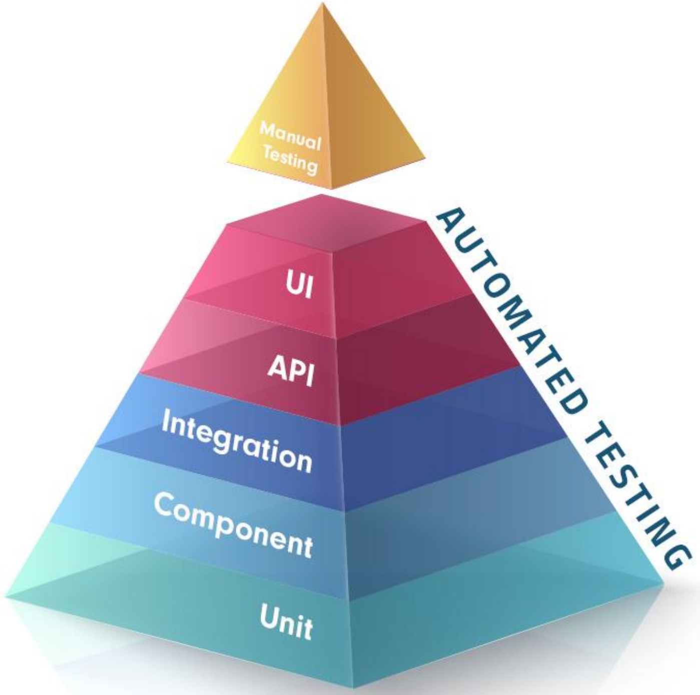
- Peu de bugs
- Lisible par un autre développeur
- Pas trop dur à faire évoluer


HOW TO WRITE GOOD CODE:
START PROJECT.
DO THINGS RIGHT OR DO THEM FAST?
FAST
CODE FAST
DOES IT WORK YET?
NO
ALMOST, BUT IT'S BECOME A MASS OF KLUDGES AND SPAGHETTI CODE.
RIGHT
CODE WELL.
ARE YOU DONE YET?
NO
NO, AND THE REQUIREMENTS HAVE CHANGED.
THROW IT ALL OUT AND START OVER.
?
GOOD CODE

{RIVIERADEV}

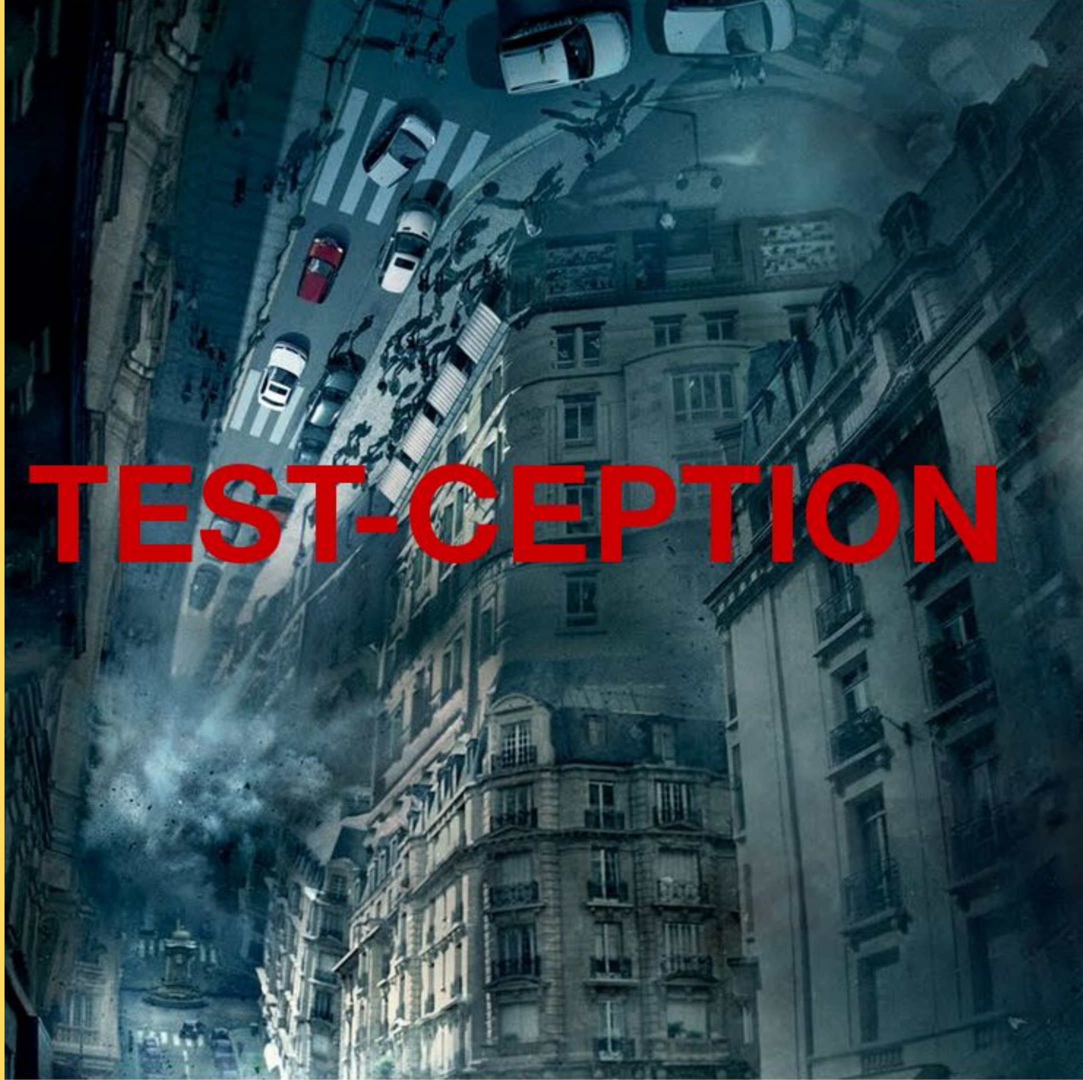

TESTS
YOU SHOULD WRITE
memegenerator.net

Stratégies de test



Du code robuste grâce aux tests

Tester les tests

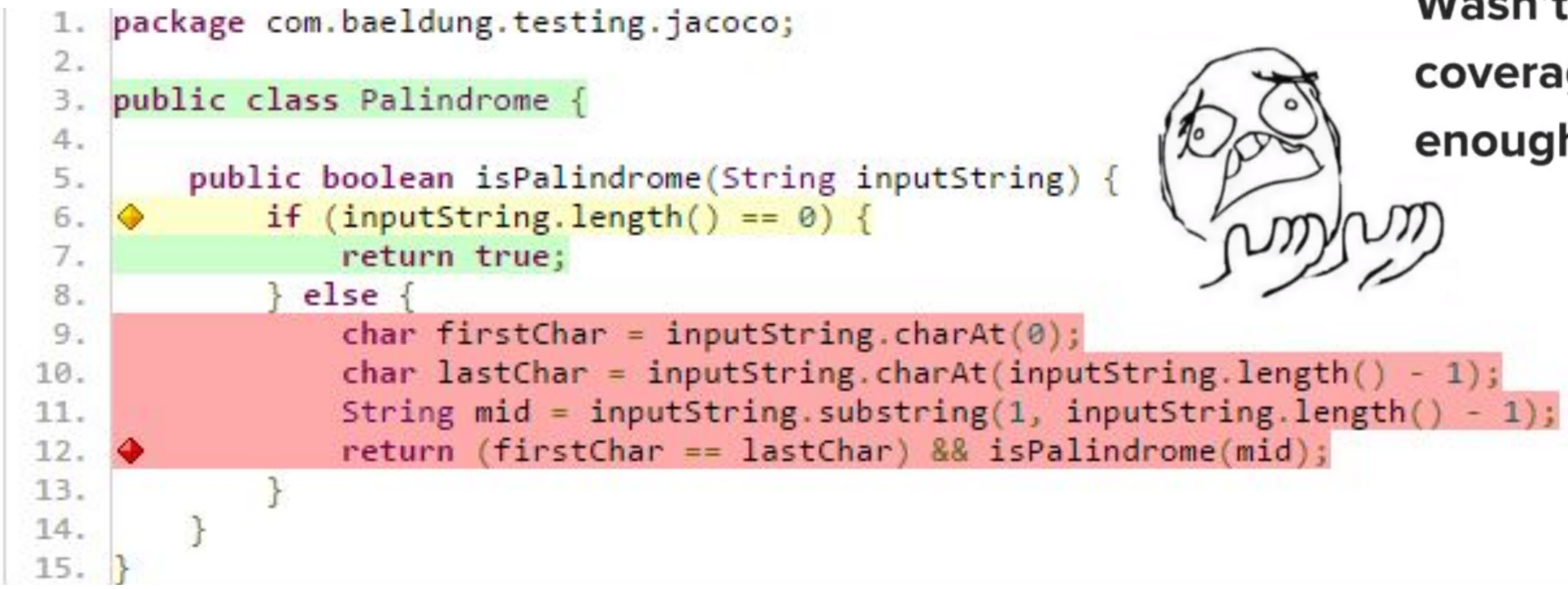


RIVIERADEV

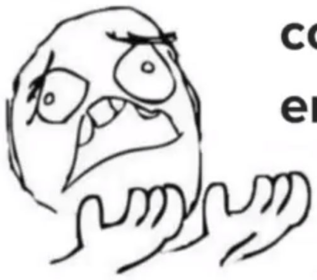
Etape 1: l’intuition

Etape 2: couverture de code

Solution 2: couverture de code



```
package com.baeldung.testing.jacoco;

public class Palindrome {

    public boolean isPalindrome(String inputString) {
        if (inputString.length() == 0) {
            return true;
        } else {
            char firstChar = inputString.charAt(0);
            char lastChar = inputString.charAt(inputString.length() - 1);
            String mid = inputString.substring(1, inputString.length() - 1);
            return (firstChar == lastChar) && isPalindrome(mid);
        }
    }
}
```



Wasn't code coverage enough?

Code exécuté par des tests != code testé

```scala
class Cart(size: Int) {
  val items = mutable.ArrayBuffer[String]()

  def add(item: String): Boolean = {
    println(s"item add: $item")
    val exists = items.contains(item)
    if(items.length < size) {
      items.append(item)
    }
    exists
  }
}
```

```scala
it("has no assert") {
 new Cart(3).add("shoes")
}
```

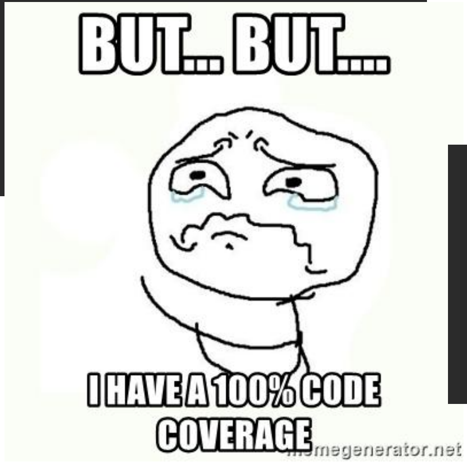
Code exécuté par des tests != code testé

```scala
class Cart(size: Int) {
  val items = mutable.ArrayBuffer[String]()

  def add(item: String): Boolean = {
    println(s"item add: $item")
    val exists = items.contains(item)
    if(items.length < size) {
      items.append(item)
    }
    exists
  }
}
```

```scala
it("has irrelevant assert") {
 new Cart(3).add("shoes") shouldBe false
}
```

Code exécuté par des tests != code testé

```
class Cart(size: Int) {
  val items = mutable.ArrayBuffer[String]()

  def add(item: String): Boolean = {
    println(s"item add: $item")
    val exists = items.contains(item)
    if(items.length < size) {
      items.append(item)
    }
    exists
  }
}
```

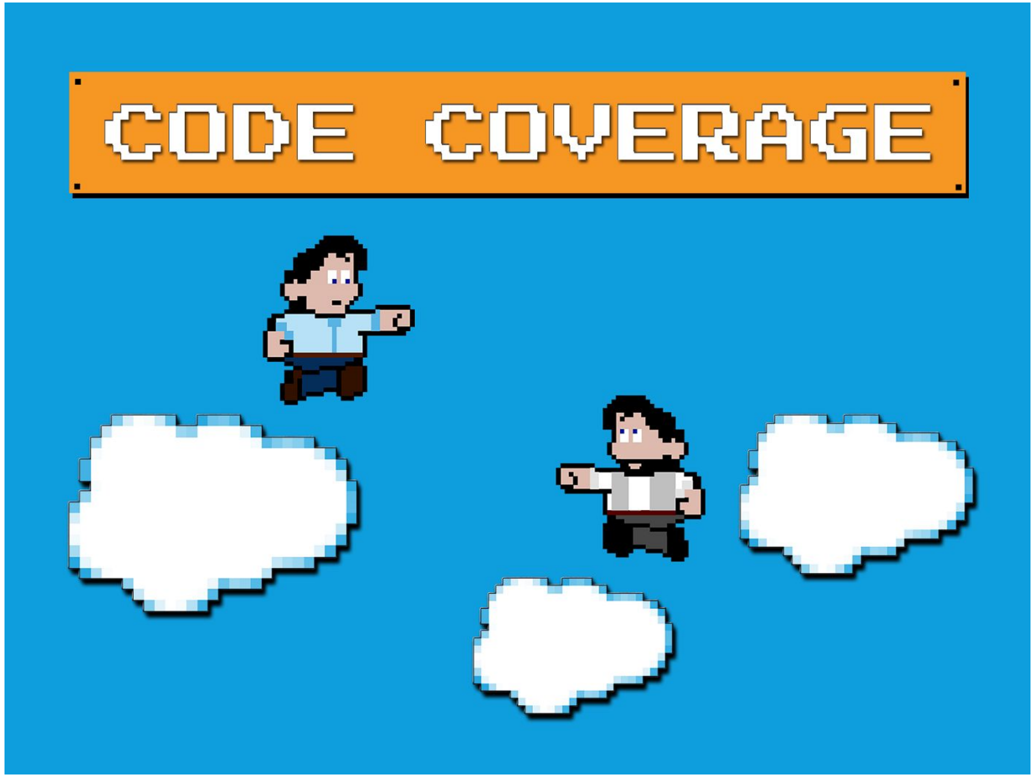


Non testé :

- les effets de bords
- la condition limite
- l'ajout dans la liste

```
it("asserts few things") {
 val cart = new Cart(3)
 cart.add("shoes")
 cart.items.length shouldBe 1
}
```

Code exécuté par des tests != code testé



Tout le code ne se vaut pas

Etape 3

Mutation testing

Mise en place: Java

```
// pom.xml
<build>
   <plugins>
       <plugin>
           <groupId>org.pitest</groupId>
           <artifactId>pitest-maven</artifactId>
           <version>1.2.4</version>
       </plugin>
   </plugins>
</build>
```

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

Mise en place: Scala

```
// project/plugins.sbt
addSbtPlugin("io.github.sugakandrey" % "sbt-scalamu" % "0.1.1")
```

Scalamu

```
$ sbt mutationTest
```

Mise en place: JavaScript

```
$ npm install stryker stryker-api stryker-html-reporter
stryker-javascript-mutator stryker-jest-runner --save-dev
```

```
// stryker.conf.js
module.exports = function(config) {
 config.set({
   testRunner: "jest",
   mutator: "javascript",
   transpilers: [],
   reporter: ["html", "progress"],
   coverageAnalysis: "all",
   mutate: ["src/**/*.js"]
 });
};
```

```
$ ./node_modules/.bin/stryker run
```

Stryker

Et plein d'autres…

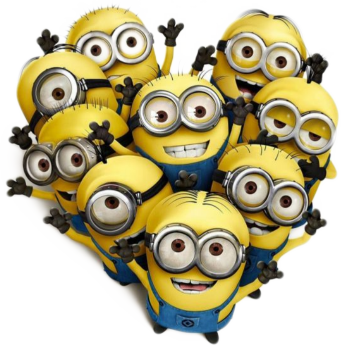
NinjaTurtles pour **C#**

Mutmut pour **Python**

mutant pour **Ruby**

Infection pour **PHP**

…



Mutation testing

Génère un mutant

Lance les tests

Vérifie le résultat

Recommence

{RIVIERADEV}

Mutant tué

Si les tests **échouent**

Le code muté a été détecté

Il est donc **correctement testé**

Mutant vivant

Si les tests **réussissent**

Le code muté n’a **pas** été détecté

Les tests sont donc insuffisants

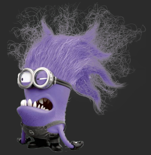
Qu’est-ce qu’un mutant ?

```
def add(item: String): Boolean = {
  println(s"item add: $item")
  val exists = items.contains(item)
  if (items.length < size) {
    items.append(item)
  }
  exists
}
```

Code original

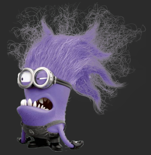
Supprime un appel de fonction

```
def add(item: String): Boolean = {
  ()
  val exists = items.contains(item)
  if (items.length < size) {
    items.append(item)
  }
  exists
}
```



Condition toujours vraie

```
def add(item: String): Boolean = {
  println(s"item add: $item")
  val exists = items.contains(item)
  if (true) {
    items.append(item)
  }
  exists
}
```



Mutations: conditions

Modification :

`<` ⇔ `<=`

`>` ⇔ `>=`

`&&` ⇔ `||`

Constante :

`true`

`false`

Inversion :

`==` ⇔ `!=`

`<` ⇔ `>=`

`>` ⇔ `<=`

`cond` ⇔ `!cond`

```java
if (items.size() < size) {
   items.add(item);
}
```

⇒

```java
if (items.size() <= size) {
   items.add(item);
}
```

Mutations: opération mathématique

Opérations :

`x++`	⇔	`x--`
`+`	⇔	`-`
`*`	⇔	`/`
`%`	⇔	`*`

Opérations binaires :

`&`	⇔	`|`
`^`	⇔	`&`
`>>`	⇔	`<<`

```
return total - discount;
```

```
return total + discount;
```

{RIVIERADEV}

Loïc Knuchel - @loicknuchel

Mutations: constantes

Change une constante :

```
true ⇔ false

   0 ⇔ 1

   x ⇔ x + 1

   x ⇔ null
```

Remplace une variable par une constante :

```
x ⇔ true / false

x ⇔ 0 / 1 / 2
```

```
return exists;
```

→

```
if(exists == null)
throw new RuntimeException();
else return null;
```

Mutations: supprimer une fonction

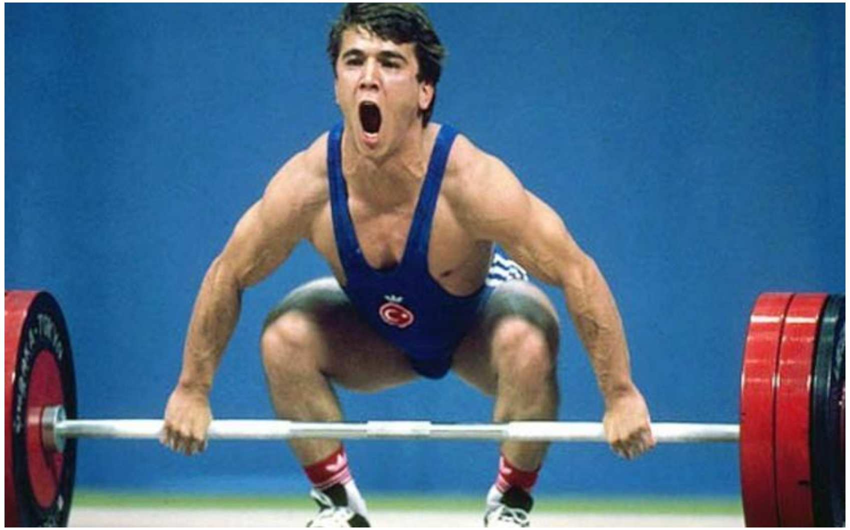
```
println(s"item add: $item")
```

→

```
```

Mutations

En pratique


Brute force

- mutants uniquement pour le code couvert par les tests
- lance uniquement les tests qui couvrent le code muté
- fonctionne en mode itératif
- à mettre en priorité pour le code critique
- activer que les mutations intéressantes

Exemple

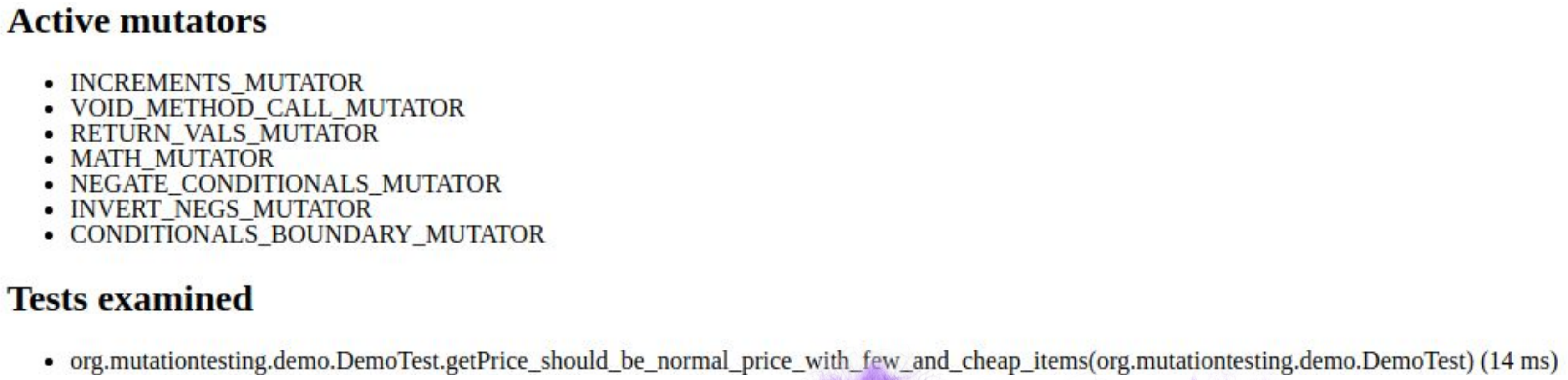
```java
/**
 * Take a list of item prices and calculate the bill :
 * - if total is higher than 50, apply 10% overall discount
 * - if more than 5 items, apply 100% discount on cheapest one
 * - if many discount apply, use the higher one
 */
public static Double getPrice(List<Double> prices) {
  Double total = sum(prices);
  Double discount = 0.0;
  if (total >= 50) {
    discount = total * 0.1;
  }
  if (prices.size() >= 5) {
    Double minPrice = min(prices);
    if (minPrice > discount) {
      discount = minPrice;
    }
  }
  return total - discount;
}
```

{RIVIERADEV}

Loïc Knuchel - @loicknuchel

Test 1

```java
@Test
public void getPrice_should_be_normal_price_with_few_and_cheap_items() {
  assertEquals(24, Demo.getPrice(Arrays.asList(4, 7, 1, 12)), 0.001);
}
```



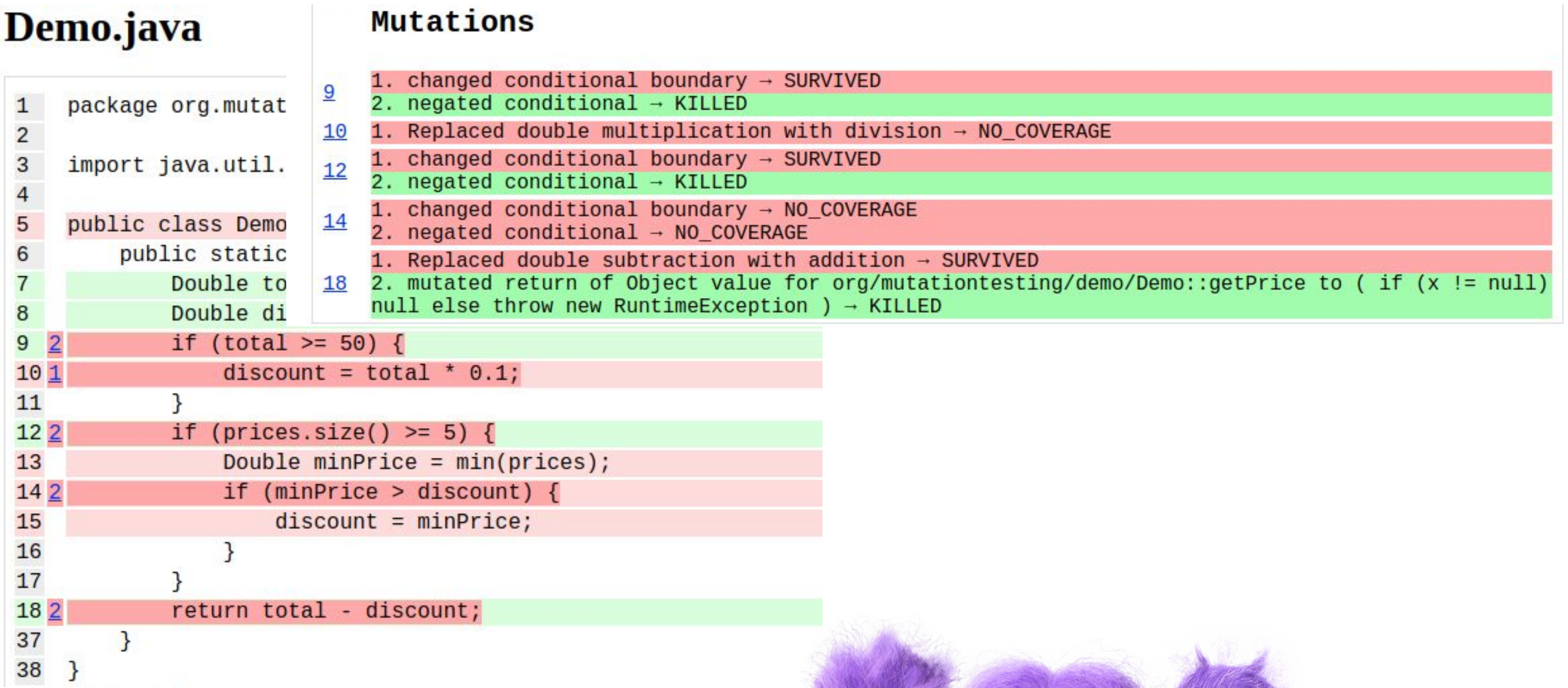
Active mutators

- INCREMENTS_MUTATOR
- VOID_METHOD_CALL_MUTATOR
- RETURN_VALS_MUTATOR
- MATH_MUTATOR
- NEGATE_CONDITIONALS_MUTATOR
- INVERT_NEGS_MUTATOR
- CONDITIONALS_BOUNDARY_MUTATOR

Tests examined

- org.mutationtesting.demo.DemoTest.getPrice_should_be_normal_price_with_few_and_cheap_items(org.mutationtesting.demo.DemoTest) (14 ms)

Test 1



Demo.java

```
1      package org.mutat
2
3      import java.util.
4
5      public class Demo
6          public static
7              Double to
8              Double di
9  2           if (total >= 50) {
10 1               discount = total * 0.1;
11             }
12 2           if (prices.size() >= 5) {
13                 Double minPrice = min(prices);
14 2               if (minPrice > discount) {
15                     discount = minPrice;
16                 }
17             }
18 2           return total - discount;
37         }
38     }
```

Mutations

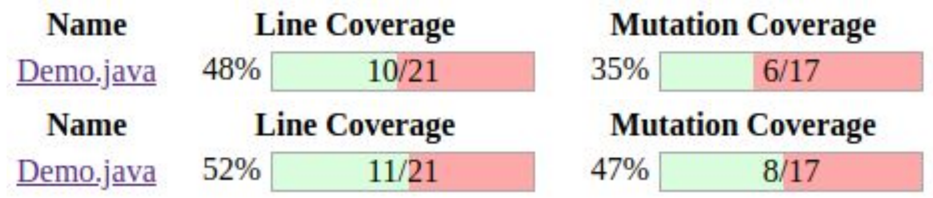
```
9   1. changed conditional boundary → SURVIVED
    2. negated conditional → KILLED
10  1. Replaced double multiplication with division → NO_COVERAGE
12  1. changed conditional boundary → SURVIVED
    2. negated conditional → KILLED
14  1. changed conditional boundary → NO_COVERAGE
    2. negated conditional → NO_COVERAGE
18  1. Replaced double subtraction with addition → SURVIVED
    2. mutated return of Object value for org/mutationtesting/demo/Demo::getPrice to ( if (x != null)
    null else throw new RuntimeException ) → KILLED
```

{RIVIERADEV}

Loïc Knuchel - @loicknuchel

Test 1 + 2

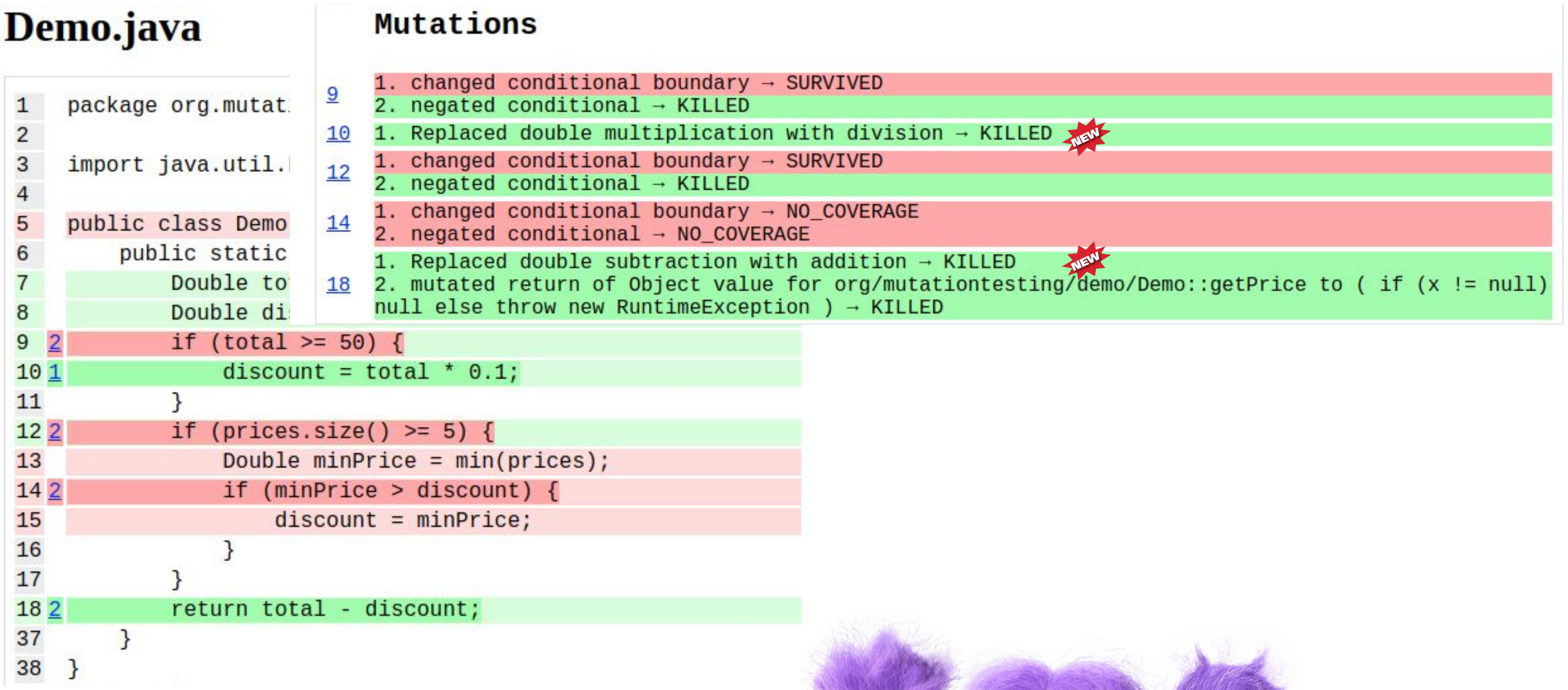
```java
@Test
public void getPrice_should_be_get_10pc_discound_on_expensive_items() {
  assertEquals(54, Demo.getPrice(Arrays.asList(10, 20, 30)), 0.001);
}
```



Name	Line Coverage		Mutation Coverage	
Demo.java	48%	10/21	35%	6/17

Name	Line Coverage		Mutation Coverage	
Demo.java	52%	11/21	47%	8/17

Test 1 + 2



Demo.java

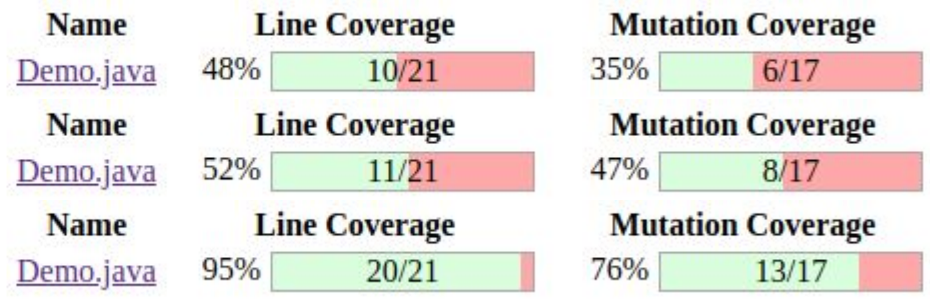
```
1       package org.mutat
2
3       import java.util.
4
5       public class Demo
6           public static
7               Double to
8               Double di
9   2           if (total >= 50) {
10  1               discount = total * 0.1;
11              }
12  2           if (prices.size() >= 5) {
13                  Double minPrice = min(prices);
14  2               if (minPrice > discount) {
15                      discount = minPrice;
16                  }
17              }
18  2           return total - discount;
37          }
38      }
```

Mutations

```
9   1. changed conditional boundary → SURVIVED
    2. negated conditional → KILLED
10  1. Replaced double multiplication with division → KILLED
12  1. changed conditional boundary → SURVIVED
    2. negated conditional → KILLED
14  1. changed conditional boundary → NO_COVERAGE
    2. negated conditional → NO_COVERAGE
18  1. Replaced double subtraction with addition → KILLED
    2. mutated return of Object value for org/mutationtesting/demo/Demo::getPrice to ( if (x != null)
    null else throw new RuntimeException ) → KILLED
```

Test 1 + 2 + 3

```java
@Test
public void getPrice_should_be_get_one_free_item_when_buy_many() {
   assertEquals(22, Demo.getPrice(Arrays.asList(3, 5, 2, 8, 1, 4)), 0.001);
}
```

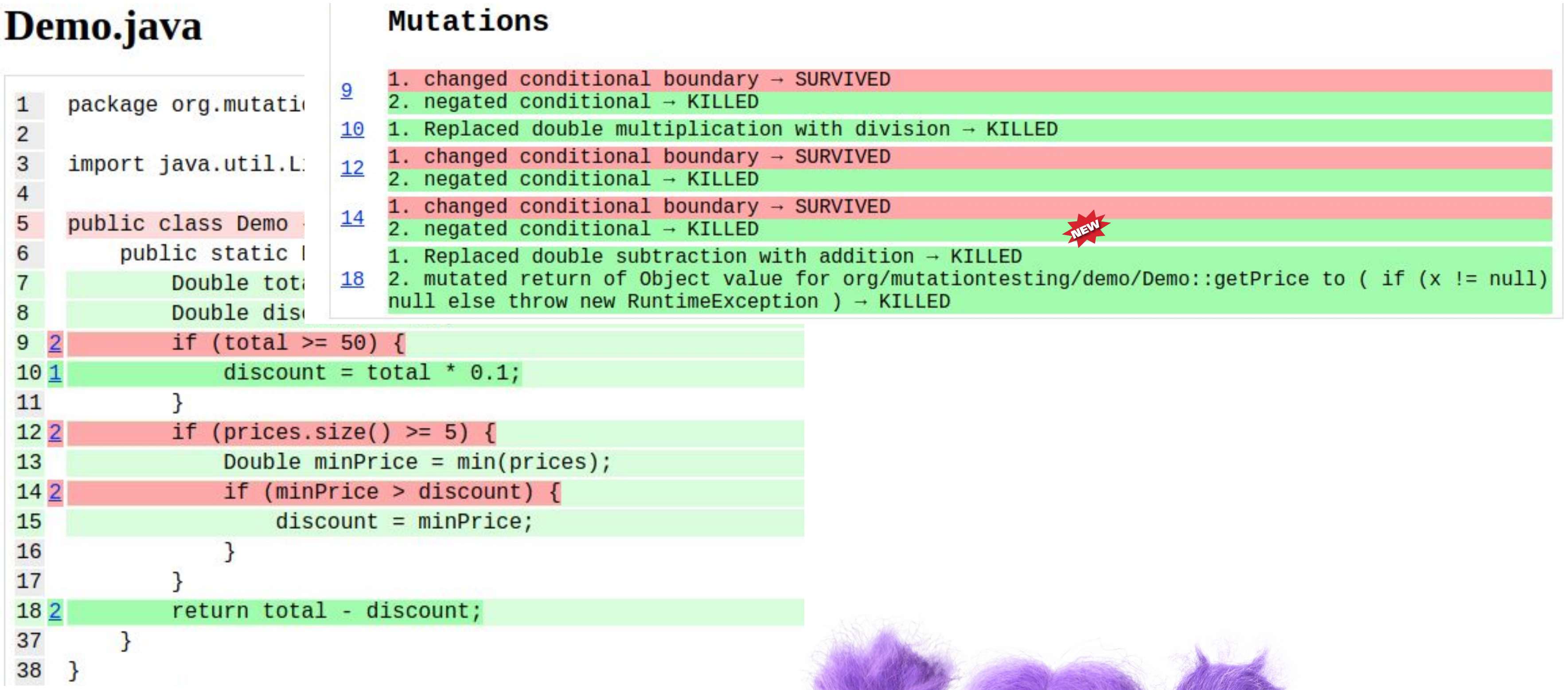


Name	Line Coverage		Mutation Coverage	
Demo.java	48%	10/21	35%	6/17

Name	Line Coverage		Mutation Coverage	
Demo.java	52%	11/21	47%	8/17

Name	Line Coverage		Mutation Coverage	
Demo.java	95%	20/21	76%	13/17

Test 1 + 2 + 3



Demo.java

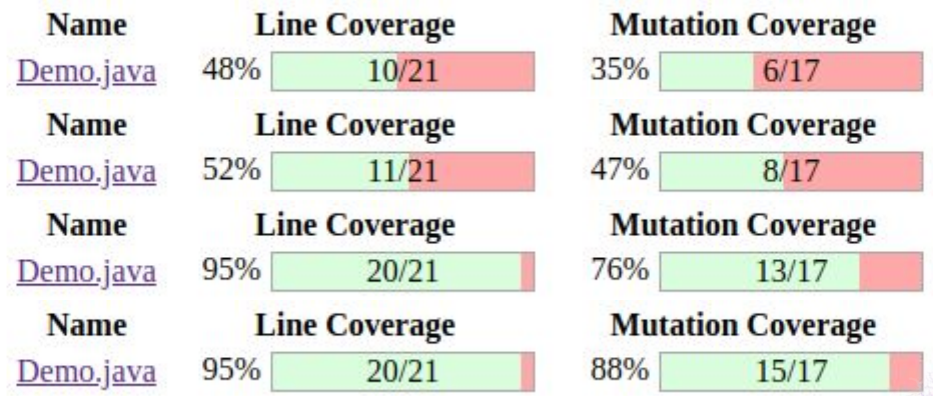
```
1      package org.mutati
2
3      import java.util.Li
4
5      public class Demo
6          public static
7              Double tota
8              Double disc
9  2           if (total >= 50) {
10 1               discount = total * 0.1;
11             }
12 2           if (prices.size() >= 5) {
13                 Double minPrice = min(prices);
14 2               if (minPrice > discount) {
15                     discount = minPrice;
16                 }
17             }
18 2           return total - discount;
37         }
38     }
```

Mutations

```
9   1. changed conditional boundary → SURVIVED
    2. negated conditional → KILLED
10  1. Replaced double multiplication with division → KILLED
12  1. changed conditional boundary → SURVIVED
    2. negated conditional → KILLED
14  1. changed conditional boundary → SURVIVED
    2. negated conditional → KILLED
18  1. Replaced double subtraction with addition → KILLED
    2. mutated return of Object value for org/mutationtesting/demo/Demo::getPrice to ( if (x != null)
    null else throw new RuntimeException ) → KILLED
```

Test 1 + 2 + 3 + 4

```
@Test
public void getPrice_should_should_test_boundary_conditions() {
  // 50 total value boundary
  assertEquals(45, Demo.getPrice(Arrays.asList(5, 10, 15, 20)), 0.001);
  // 5 item boundary
  assertEquals(43, Demo.getPrice(Arrays.asList(7, 8, 15, 10, 10)), 0.001);
}
```



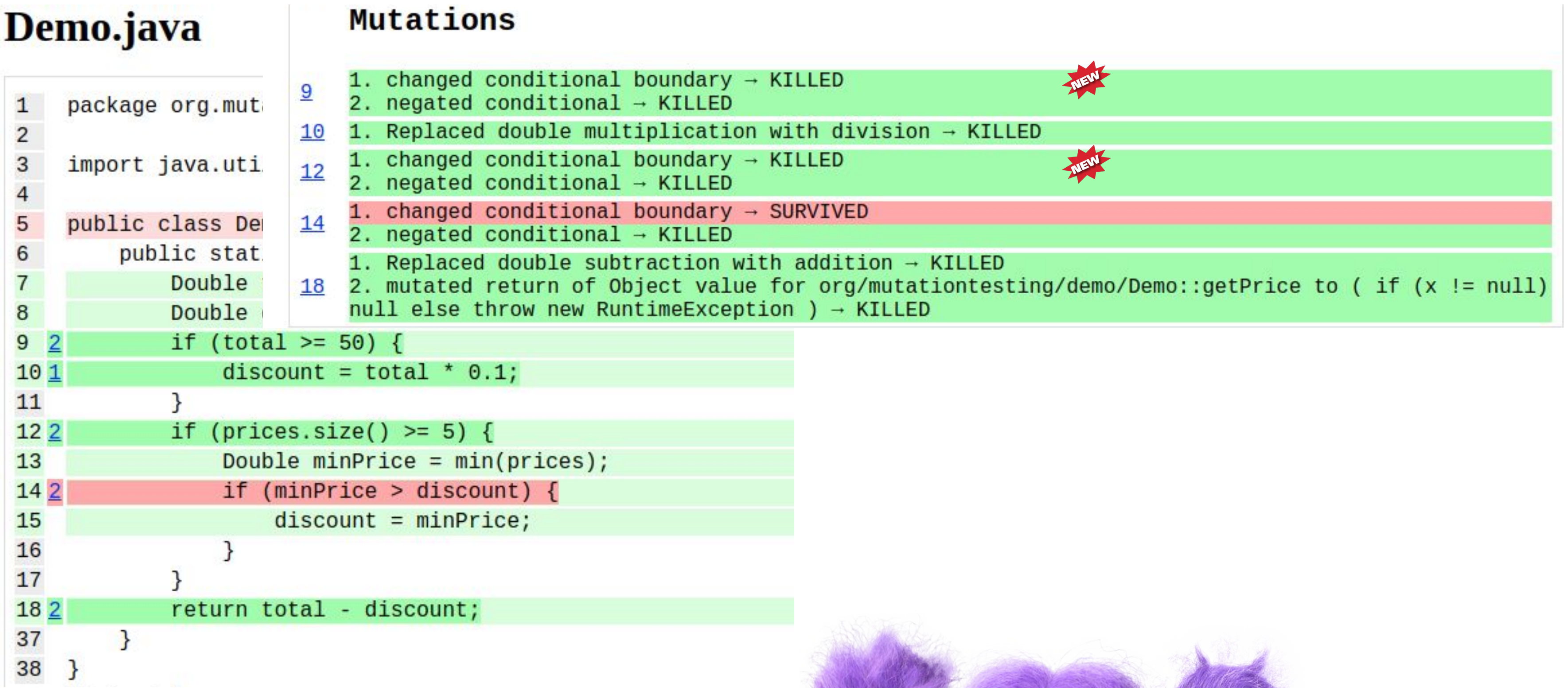
Name	Line Coverage		Mutation Coverage	
Demo.java	48%	10/21	35%	6/17

Name	Line Coverage		Mutation Coverage	
Demo.java	52%	11/21	47%	8/17

Name	Line Coverage		Mutation Coverage	
Demo.java	95%	20/21	76%	13/17

Name	Line Coverage		Mutation Coverage	
Demo.java	95%	20/21	88%	15/17

Test 1 + 2 + 3 + 4



Demo.java

```
1      package org.mut
2
3      import java.uti
4
5      public class De
6          public stat
7              Double
8              Double
9  2           if (total >= 50) {
10 1               discount = total * 0.1;
11             }
12 2           if (prices.size() >= 5) {
13                 Double minPrice = min(prices);
14 2               if (minPrice > discount) {
15                     discount = minPrice;
16                 }
17             }
18 2           return total - discount;
37         }
38     }
```

Mutations

```
9   1. changed conditional boundary → KILLED
    2. negated conditional → KILLED
10  1. Replaced double multiplication with division → KILLED
12  1. changed conditional boundary → KILLED
    2. negated conditional → KILLED
14  1. changed conditional boundary → SURVIVED
    2. negated conditional → KILLED
18  1. Replaced double subtraction with addition → KILLED
    2. mutated return of Object value for org/mutationtesting/demo/Demo::getPrice to ( if (x != null)
    null else throw new RuntimeException ) → KILLED
```

{RIVIERADEV}

Code source

https://github.com/loicknuchel/mutation-testing-sample

Java / Scala / JavaScript / PR acceptées ;)

Conclusion

Facile à mettre en place

Impossible à tuer

Tester ses tests

Couplage code ⇔ tests

Long à exécuter

Impossible à contourner

Meilleure couverture de code !

Questions ?

Slides:
http://bit.ly/rivieradev-2018-mutation-testing