Softwerkskammer
Software Craftsmanship Communities

Tests und Code verbessern mit Mutation Testing

Hands-On Coding
Softwerkskammer
Nürnberg

Beyond Code Coverage

- Unsere Tests prüfen unseren Code
- Wer prüft unsere Tests?

- Hohe Code Coverage => Hohe Güte der Tests?
- 100% Code Coverage, aber keine Assertions …

Mutation Testing

Einfache Grundidee:

- <code>**SOLID**</code> original
- Alle Tests grün als Voraussetzung
- <code>**SOLD**</code> mit Mutation
- Alle Tests immer noch grün => Mutation „überlebt“
- Mindestens ein Test rot => Mutation „getötet“
- Güte der Tests = Anzahl getöteter Mutationen / Anzahl aller Mutationen

Beispiele für Mutatoren

- Relationale Operatoren ersetzen
    - Mit anderen Grenzen if (a **>=** b) => if (a **>** b)
    - Durch das Gegenteil if (a **>=** b) => if (a **<** b)
    - Durch Konstanten if (a **>=** b) => if (**true**)
- Arithmetische Operatoren ersetzen
    - a = b **+** c => a = b **-** c
- Rückgabe-Werte verändern
    - return **x** => return **x != null ? null : throw new RuntimeException()**
- Anweisungen entfernen
    - ~~**doSomething()**~~

Probleme

- Alle Tests müssen für Original-Code grün sein
    - Nicht immer und überall selbstverständlich …
- Endlosschleifen durch Mutation erzeugt
    - Abbruch durch Timeout für jeden Test
- Laufzeit der Tests im Rahmen halten
    - Nur bestimmte Tests ausführen
    - Zu mutierende Klassen und Tests einschränken
- Semantisch äquivalente Mutationen erkennen
    - Schwierig!

Werkzeug-Auswahl

Java - PIT http://pitest.org

JavaScript – Stryker https://stryker-mutator.github.io

C# - Visual Mutator https://visualmutator.github.io/web

Ruby – Mutant https://github.com/mbj/mutant

PHP – Humbug https://github.com/humbug/humbug

Python – Cosmic Ray http://cosmic-ray.readthedocs.io

Hands-On

Übungsprojekt

- Domain: Payment Authorisation

Alternativ

- Sprache: Java
- Werkzeug: PIT

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

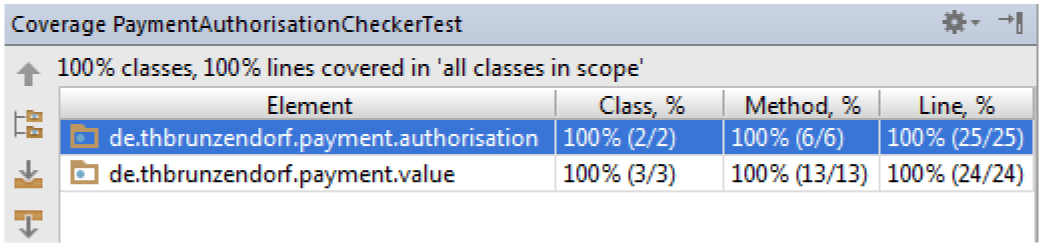
Oder

- Sprache: JavaScript
- Werkzeug: Stryker

https://github.com/thbrunzendorf/mutation-testing-js

Hands-On PIT

- Basis Code Coverage 100%



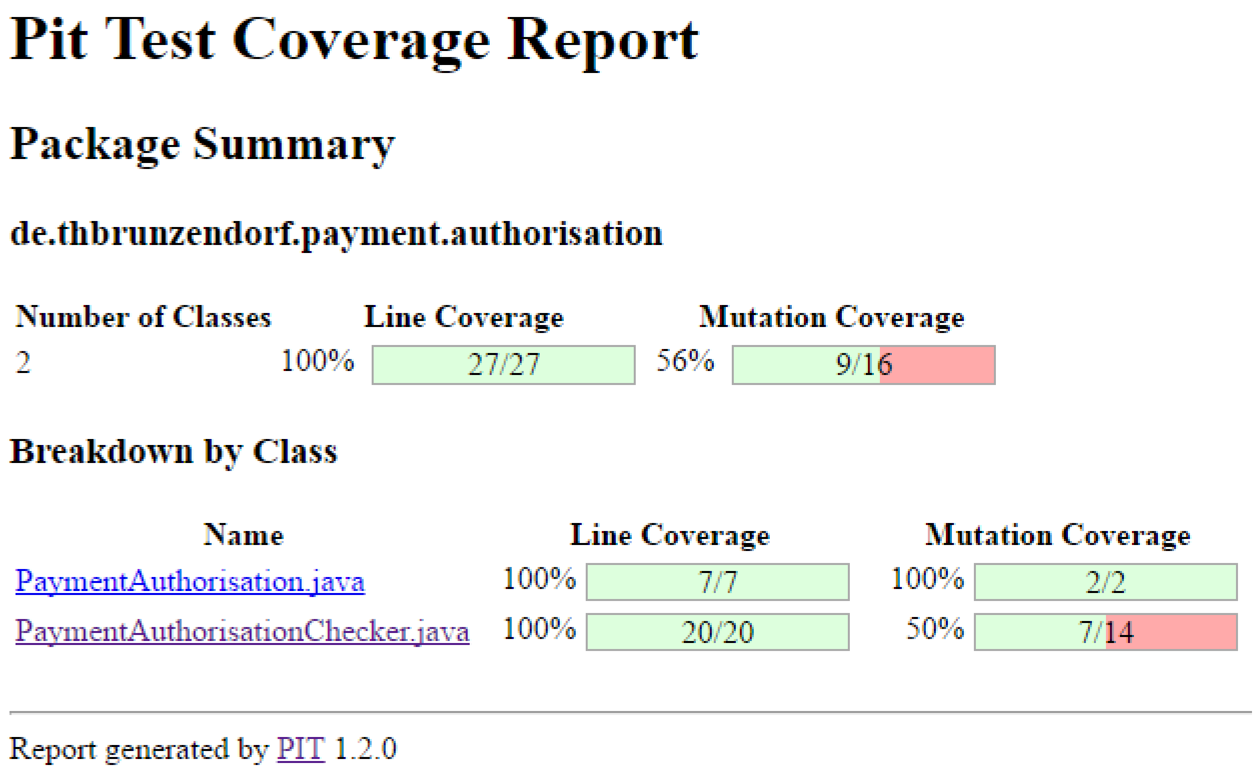
Coverage PaymentAuthorisationCheckerTest

100% classes, 100% lines covered in 'all classes in scope'

Element	Class, %	Method, %	Line, %
de.thbrunzendorf.payment.authorisation	100% (2/2)	100% (6/6)	100% (25/25)
de.thbrunzendorf.payment.value	100% (3/3)	100% (13/13)	100% (24/24)

- mvn clean install
- mvn org.pitest:pitest-maven:mutationCoverage
- Ergebnis unter target/pit-reports/…

Hands-On PIT



Pit Test Coverage Report

Package Summary

de.thbrunzendorf.payment.authorisation

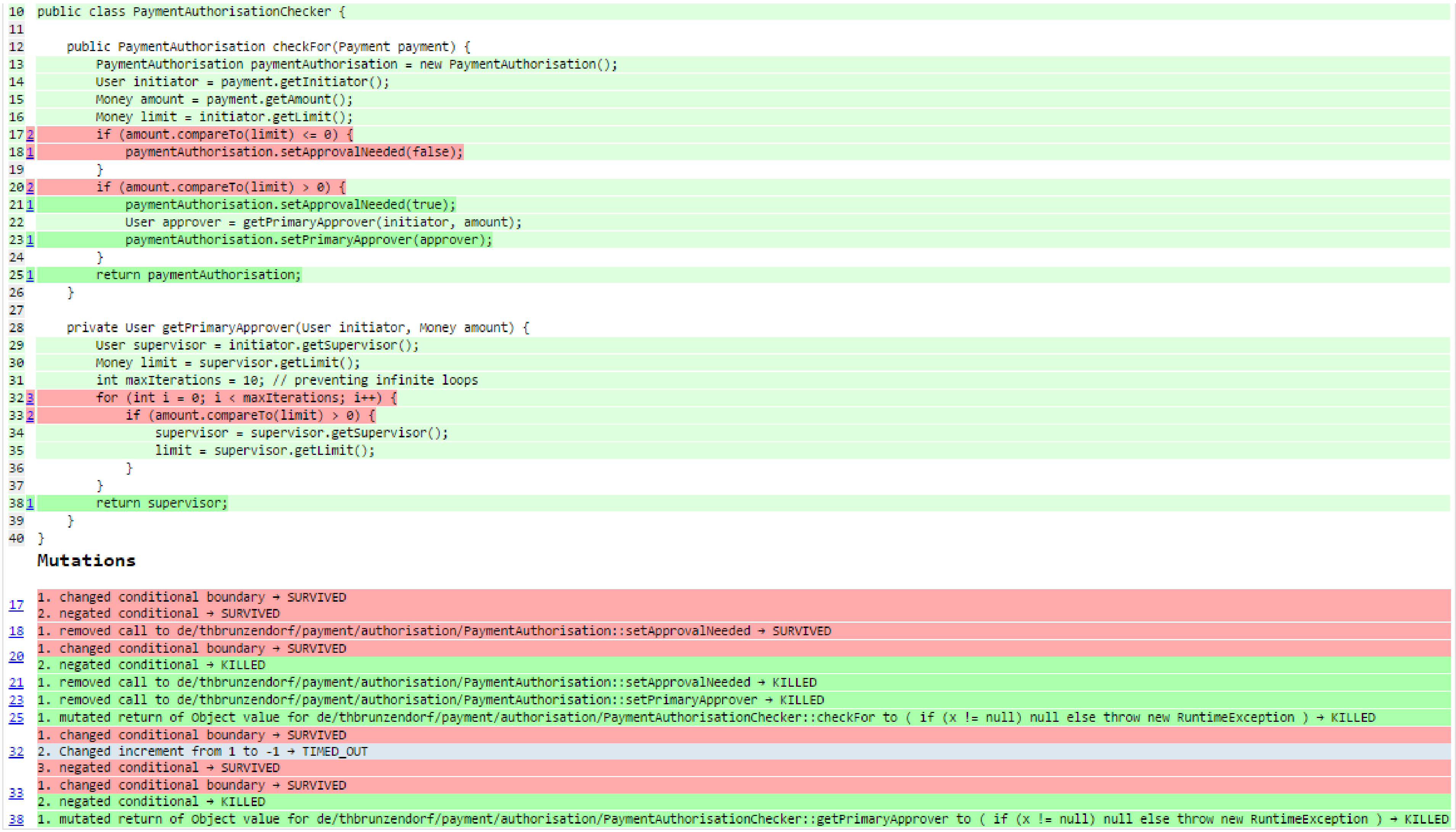
Number of Classes	Line Coverage		Mutation Coverage	
2	100%	27/27	56%	9/16

Breakdown by Class

Name	Line Coverage		Mutation Coverage	
PaymentAuthorisation.java	100%	7/7	100%	2/2
PaymentAuthorisationChecker.java	100%	20/20	50%	7/14

Report generated by PIT 1.2.0

Hands-On PIT



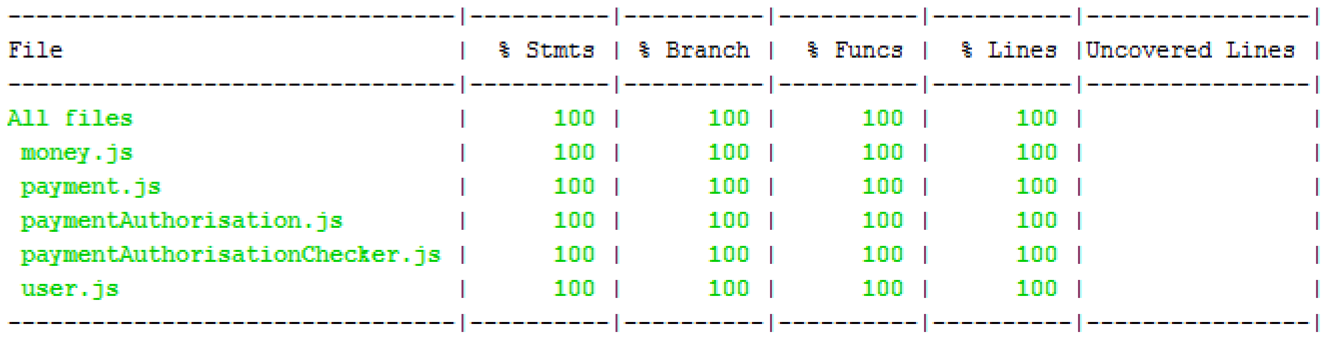
```
10   public class PaymentAuthorisationChecker {
11
12       public PaymentAuthorisation checkFor(Payment payment) {
13           PaymentAuthorisation paymentAuthorisation = new PaymentAuthorisation();
14           User initiator = payment.getInitiator();
15           Money amount = payment.getAmount();
16           Money limit = initiator.getLimit();
17 2         if (amount.compareTo(limit) <= 0) {
18 1             paymentAuthorisation.setApprovalNeeded(false);
19           }
20 2         if (amount.compareTo(limit) > 0) {
21 1             paymentAuthorisation.setApprovalNeeded(true);
22               User approver = getPrimaryApprover(initiator, amount);
23 1             paymentAuthorisation.setPrimaryApprover(approver);
24           }
25 1         return paymentAuthorisation;
26       }
27
28       private User getPrimaryApprover(User initiator, Money amount) {
29           User supervisor = initiator.getSupervisor();
30           Money limit = supervisor.getLimit();
31           int maxIterations = 10; // preventing infinite loops
32 3         for (int i = 0; i < maxIterations; i++) {
33 2             if (amount.compareTo(limit) > 0) {
34                   supervisor = supervisor.getSupervisor();
35                   limit = supervisor.getLimit();
36               }
37           }
38 1         return supervisor;
39       }
40   }
```

Mutations

```
17  1. changed conditional boundary → SURVIVED
    2. negated conditional → SURVIVED
18  1. removed call to de/thbrunzendorf/payment/authorisation/PaymentAuthorisation::setApprovalNeeded → SURVIVED
20  1. changed conditional boundary → SURVIVED
    2. negated conditional → KILLED
21  1. removed call to de/thbrunzendorf/payment/authorisation/PaymentAuthorisation::setApprovalNeeded → KILLED
23  1. removed call to de/thbrunzendorf/payment/authorisation/PaymentAuthorisation::setPrimaryApprover → KILLED
25  1. mutated return of Object value for de/thbrunzendorf/payment/authorisation/PaymentAuthorisationChecker::checkFor to ( if (x != null) null else throw new RuntimeException ) → KILLED
32  1. changed conditional boundary → SURVIVED
    2. Changed increment from 1 to -1 → TIMED_OUT
    3. negated conditional → SURVIVED
33  1. changed conditional boundary → SURVIVED
    2. negated conditional → KILLED
38  1. mutated return of Object value for de/thbrunzendorf/payment/authorisation/PaymentAuthorisationChecker::getPrimaryApprover to ( if (x != null) null else throw new RuntimeException ) → KILLED
```

Hands-On Stryker

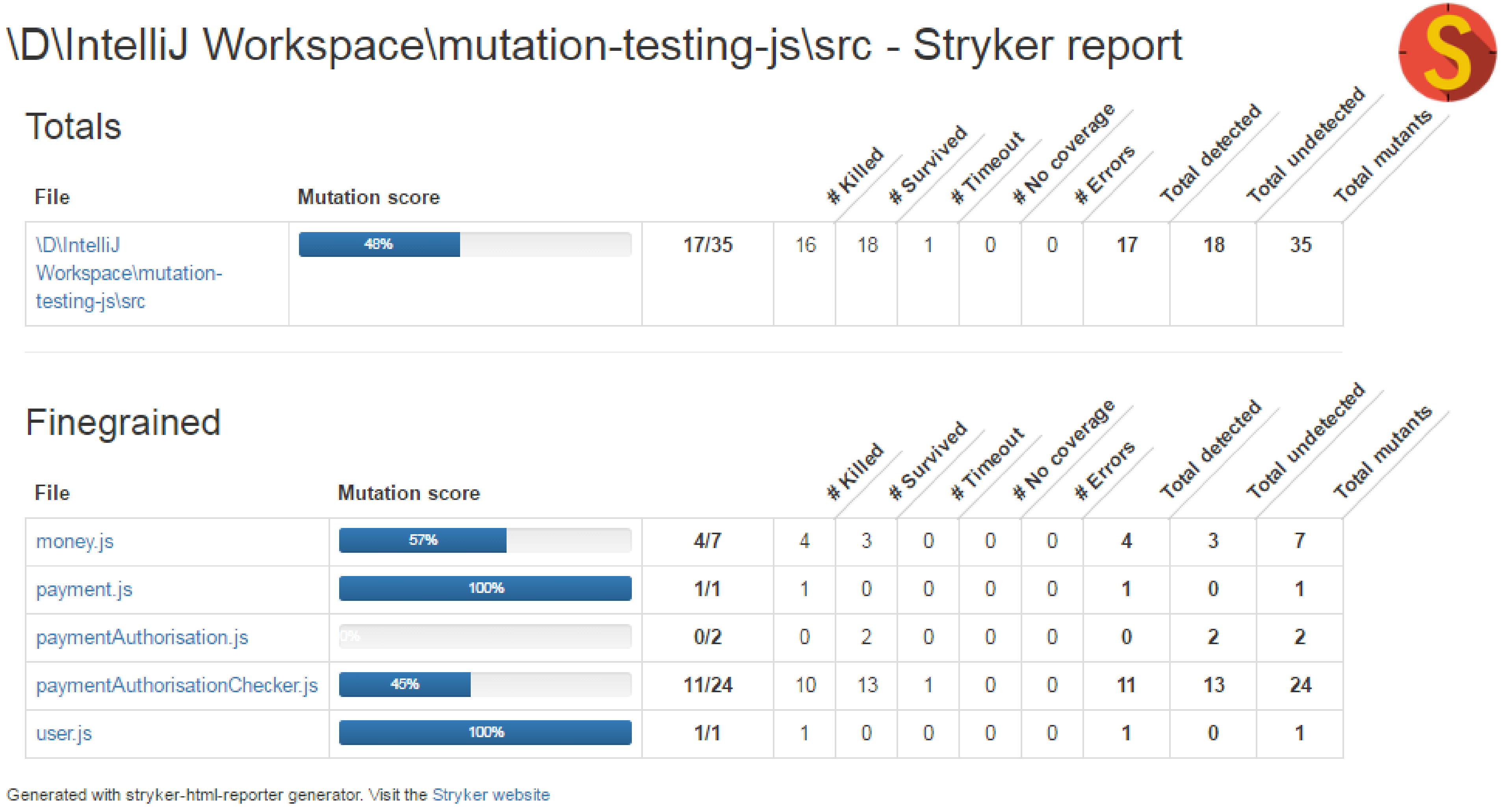
- Basis Code Coverage 100%



```
--------------------------------|----------|----------|----------|----------|----------------|
File                            |  % Stmts | % Branch |  % Funcs |  % Lines |Uncovered Lines |
--------------------------------|----------|----------|----------|----------|----------------|
All files                       |      100 |      100 |      100 |      100 |                |
 money.js                       |      100 |      100 |      100 |      100 |                |
 payment.js                     |      100 |      100 |      100 |      100 |                |
 paymentAuthorisation.js        |      100 |      100 |      100 |      100 |                |
 paymentAuthorisationChecker.js |      100 |      100 |      100 |      100 |                |
 user.js                        |      100 |      100 |      100 |      100 |                |
--------------------------------|----------|----------|----------|----------|----------------|
```

- npm run-script stryker
- Ergebnis unter reports/mutation/…

Hands-On Stryker



\D\IntelliJ Workspace\mutation-testing-js\src - Stryker report

Totals

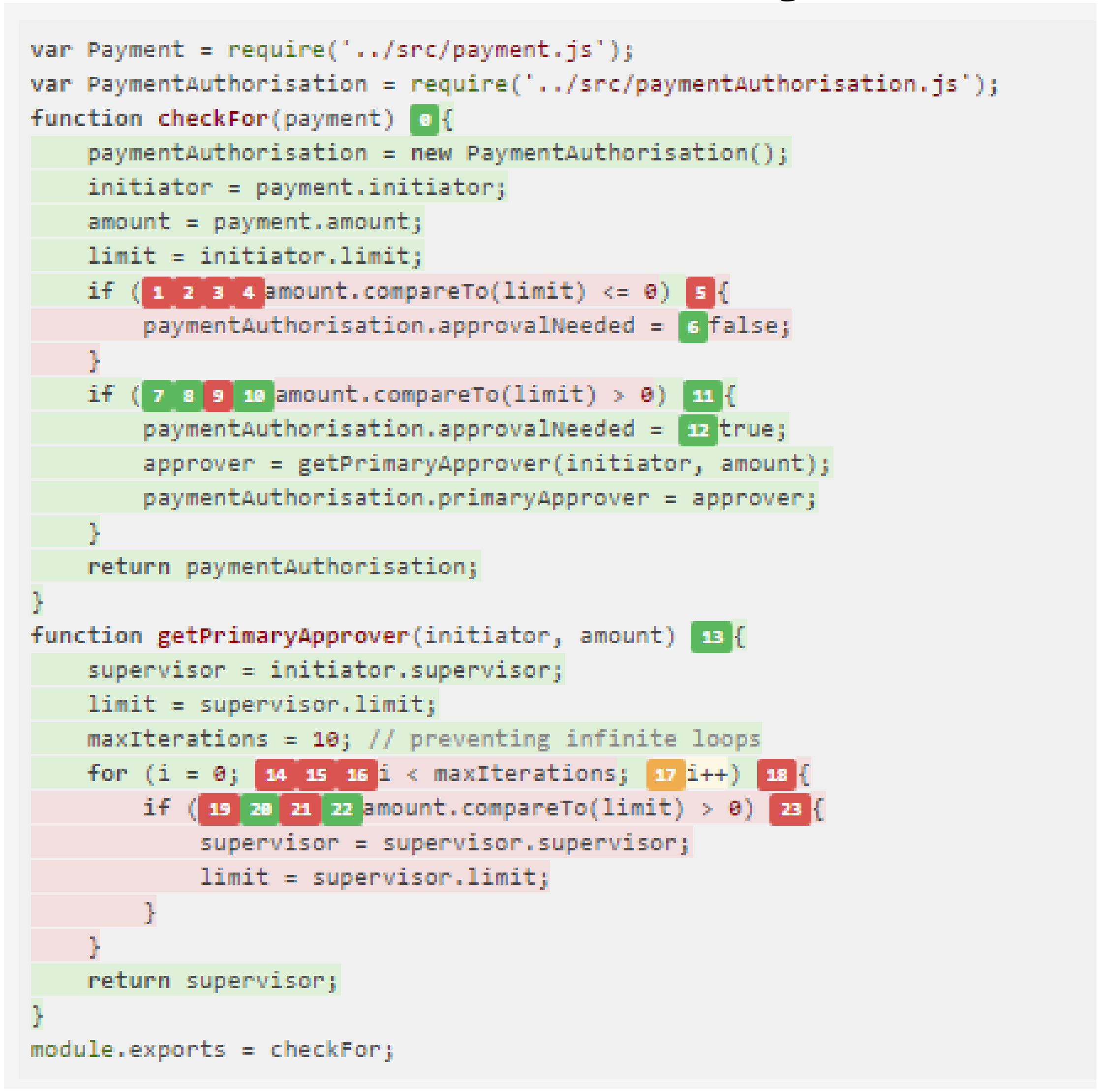
File	Mutation score		# Killed	# Survived	# Timeout	# No coverage	# Errors	Total detected	Total undetected	Total mutants
\D\IntelliJ Workspace\mutation-testing-js\src	48%	17/35	16	18	1	0	0	17	18	35

Finegrained

File	Mutation score		# Killed	# Survived	# Timeout	# No coverage	# Errors	Total detected	Total undetected	Total mutants
money.js	57%	4/7	4	3	0	0	0	4	3	7
payment.js	100%	1/1	1	0	0	0	0	1	0	1
paymentAuthorisation.js	0%	0/2	0	2	0	0	0	0	2	2
paymentAuthorisationChecker.js	45%	11/24	10	13	1	0	0	11	13	24
user.js	100%	1/1	1	0	0	0	0	1	0	1

Generated with stryker-html-reporter generator. Visit the Stryker website

Hands-On Stryker



```
var Payment = require('../src/payment.js');
var PaymentAuthorisation = require('../src/paymentAuthorisation.js');
function checkFor(payment) 0 {
    paymentAuthorisation = new PaymentAuthorisation();
    initiator = payment.initiator;
    amount = payment.amount;
    limit = initiator.limit;
    if ( 1 2 3 4 amount.compareTo(limit) <= 0) 5 {
        paymentAuthorisation.approvalNeeded = 6 false;
    }
    if ( 7 8 9 10 amount.compareTo(limit) > 0) 11 {
        paymentAuthorisation.approvalNeeded = 12 true;
        approver = getPrimaryApprover(initiator, amount);
        paymentAuthorisation.primaryApprover = approver;
    }
    return paymentAuthorisation;
}
function getPrimaryApprover(initiator, amount) 13 {
    supervisor = initiator.supervisor;
    limit = supervisor.limit;
    maxIterations = 10; // preventing infinite loops
    for (i = 0; 14 15 16 i < maxIterations; 17 i++) 18 {
        if ( 19 20 21 22 amount.compareTo(limit) > 0) 23 {
            supervisor = supervisor.supervisor;
            limit = supervisor.limit;
        }
    }
    return supervisor;
}
module.exports = checkFor;
```

Hands-On

Kill the mutants!

Diskussion

!?

Mutation Testing?

- Eine gute Code Coverage ist notwendige, aber nicht hinreichende Bedingung für gute Tests
- Mutation Testing hilft, fehlende Testfälle zu finden
  - Zusätzliche Testdaten, z.B. Grenzwerte
  - Zusätzliche Assertions
- Mutation Testing hilft, überflüssigen Code zu finden
  - Fallen uns keine Testfälle ein, die eine Mutation töten, ist der mutierte Code wahrscheinlich redundant
- Pro Tip: Testgetriebene Entwicklung reduziert die Überlebenschancen von Mutanten drastisch

Fragen?

Thorsten Brunzendorf

@thbrunzendorf

Vielen Dank!