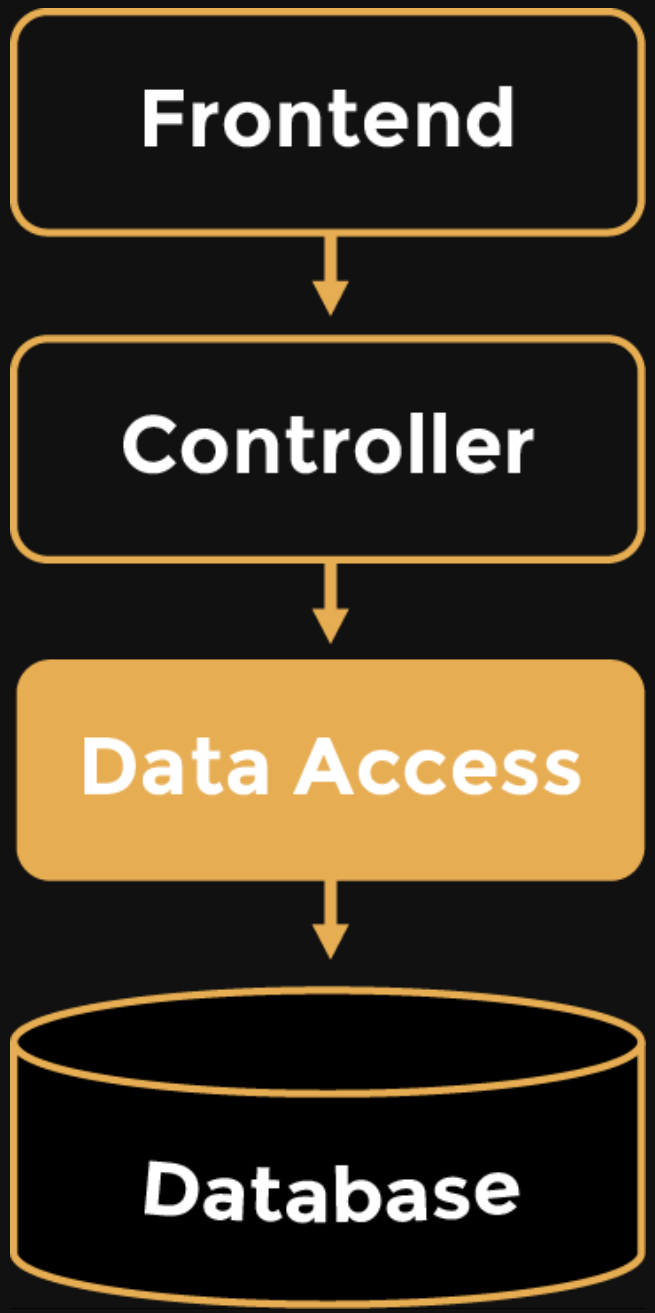
Und wer testet die Tests?

Mutationstesten mit PIT

Johannes Dienst

Warum Mutationstesten?

Legacy Projekt


Frontend
Controller
Data Access
Database

100% aussagekräftige Tests

100% Zeilenabdeckung

Wir sind fertig!

So einfach?

100% ≠ Fehlerfreiheit

⚠ Subtile Bugs ⚠

```
29      List<Integer> list = new ArrayList<>();
30      list.addAll(coll);
31
32  1   Collections.sort(list);
33  1   log(list);
34
35  1   return Collections.unmodifiableList(list);
36    }
37
38    private void log(List<Integer> list)
39    {
40  1   System.out.println(
41        list.stream().map(Object::toString)
42        .collect(Collectors.joining(", ")));
43    }
```

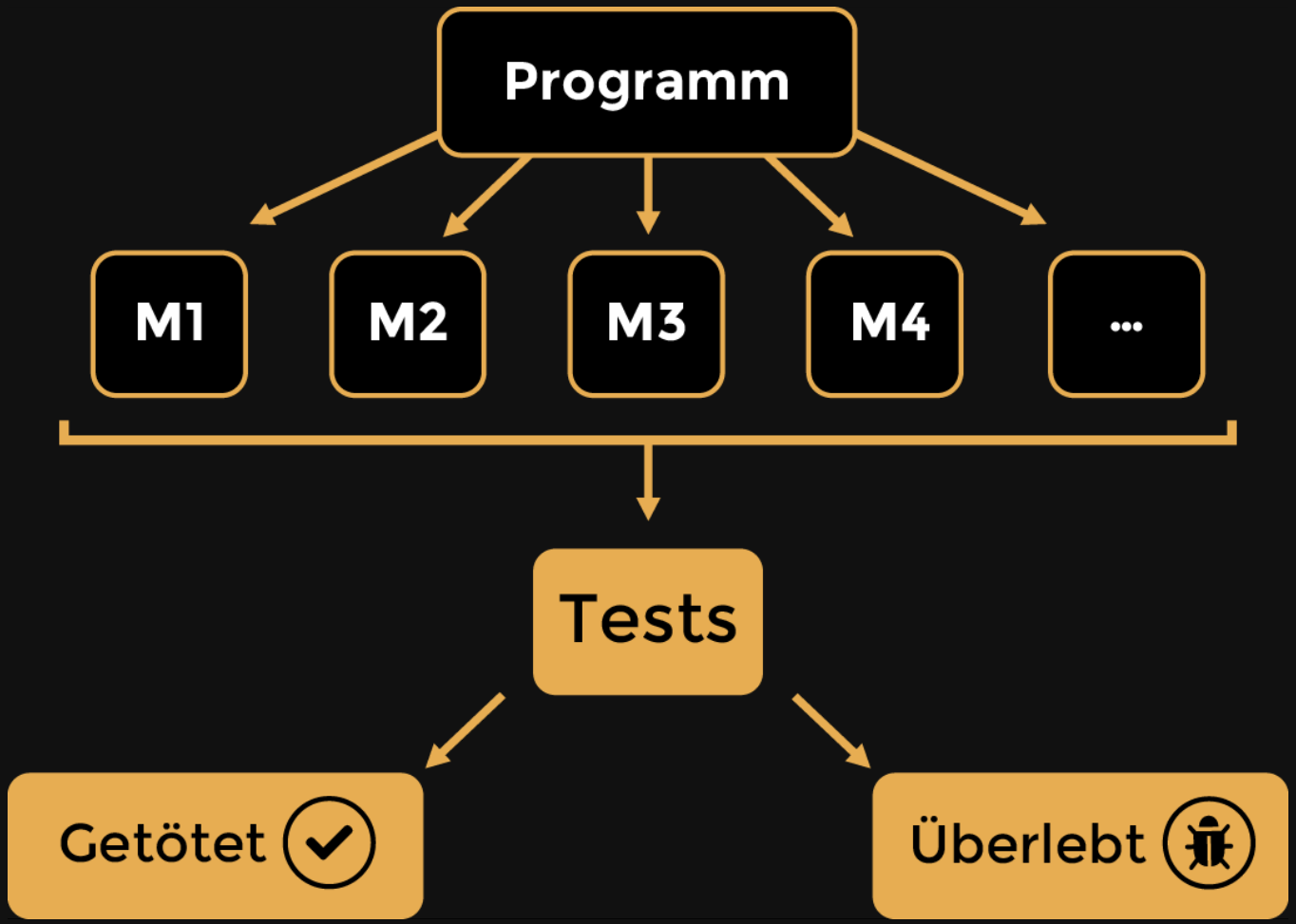
```
219     Boolean tResult = jdbcTemplate.query(
220       tQuery.toString().replace("or)", ")"),
221       new Object[] { number },
222       new ResultSetExtractor<Boolean>()
223       {
224         @Override
225         public Boolean extractData(ResultSet aResultSet) throws SQLException
226         {
227 2         if (aResultSet.next()) return Boolean.TRUE;
228           else return Boolean.FALSE;
229         }
230       } );
231     return tResult.booleanValue();
```

Tests härten!

Wie?

Mutationstesten

Richard Lipton 1971


Programm
M1
M2
M3
M4
...
Tests
Getötet
Überlebt

Gold Standard

⇨ aussagekräftige Tests

PIT

7 Default Mutatoren

12 Experimentelle Mutatoren

Default

Bedingungen, Mathematisch

Rückgabetyp

Entfernung von void-Methodenaufrufen

Schnell

ant, maven, gradle etc.

Menschenlesbare Reports

Testsuite

171 Tests

Laufzeit

Ohne PIT: 0.5 min

Mit PIT: 4 min

Erkenntnisse

Bugs

Seiteneffekte

```
205 1     if (!tResult.booleanValue())
206       {
207         jdbcTemplate.update(
208           "delete from TABLE where \"VALUE\"=?",
209           new Object[] { value });
210       }
```

False positives

```
102         UserTO tUser = (UserTO)aUserTO.clone();
103 1       tUser.setId(tOID);
104 1       tUser.setLastModificationTime(tTimestamp);
105         return tUser;
```

Demo

```
<plugin>
  <groupId>org.pitest</groupId>
  <artifactId>pitest-maven</artifactId>
  <version>1.1.11</version>
  <configuration>
    <targetClasses>
      <param>de.*</param>
    </targetClasses>
    <targetTests>
      <param>de.*</param>
    </targetTests>
    <threads>3</threads>
  </configuration>
</plugin>
```

```
public class Fibonacci {

  public int calc(int i) {
    if (i == 0) {
      return 0;
    }

    if (i <= 2) {
      return 1;
    }

    return calc(i-1) + calc(i-2);
  }
}
```

```java
@Test public void seedValue0() {
  assertEquals( 0, fib.calc( 0));
}

@Test public void seedValue1() {
  assertEquals( 1, fib.calc( 1));
}

@Test public void seedValue2() {
  assertEquals( 1, fib.calc( 2));
}

@Test public void value3() {
  assertEquals( 2, fib.calc( 3));
}

@Test public void value11() {
  assertEquals( 89, fib.calc( 11));
}
```

```
public class Sort {

  public static List sort(List coll) {
    List list = new ArrayList<>();
    list.addAll(coll);

    Collections.sort( list);
    log(list);

    return Collections.unmodifiableList( list);
  }

  private static void log(List list) {
    System.out.println(
      list.stream().map(Object::toString)
      .collect(Collectors.joining( ", ")));
  }
}
```

```java
@Test public void emptyList() {
  assertEquals( true, Sort.sort(Collections.<Integer>emptyList()).isEmpty());
}

@Test public void oneList() {
  assertEquals( false,
    Sort.sort(Stream.of( 42).collect(Collectors.toList())).isEmpty());
}

@Test public void twoList() {
  assertEquals( new Integer(1),
    Sort.sort(Stream.of( 2, 3, 1, 8).collect(Collectors.toList())).get( 0));
}
```

Pit Test Coverage Report

Project Summary

Number of Classes	Line Coverage		Mutation Coverage	
2	100%	16/16	77%	10/13

Breakdown by Package

Name	Number of Classes	Line Coverage		Mutation Coverage	
de.jdienst	2	100%	16/16	77%	10/13

Report generated by PIT 1.1.11

Pit Test Coverage Report

Package Summary

de.jdienst

Number of Classes	Line Coverage		Mutation Coverage	
2	100%	16/16	77%	10/13

Breakdown by Class

Name	Line Coverage		Mutation Coverage	
Fibonacci.java	100%	6/6	89%	8/9
Sort.java	100%	10/10	50%	2/4

Report generated by PIT 1.1.11

Fibonacci.java

```
1   package de.jdienst;
2
3   public class Fibonacci
4   {
5
6     public int calc(int i)
7     {
8   1   if (i == 0)
9       {
10  1     return 0;
11      }
12
13  2   if (i <= 2)
14      {
15  1     return 1;
16      }
17
18  4   return calc(i-1) + calc(i-2);
19    }
20
21  }
```

Mutations

8	1. negated conditional → KILLED
10	1. replaced return of integer sized value with (x == 0 ? 1 : 0) → KILLED
13	1. changed conditional boundary → SURVIVED 2. negated conditional → KILLED
15	1. replaced return of integer sized value with (x == 0 ? 1 : 0) → KILLED
18	1. Replaced integer subtraction with addition → KILLED 2. Replaced integer subtraction with addition → KILLED 3. Replaced integer addition with subtraction → KILLED 4. replaced return of integer sized value with (x == 0 ? 1 : 0) → KILLED

Active mutators

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

Tests examined

- de.jdienst.Fibonacchi_Test.value3(de.jdienst.Fibonacchi_Test) (1 ms)
- de.jdienst.Fibonacchi_Test.seedValue0(de.jdienst.Fibonacchi_Test) (8 ms)
- de.jdienst.Fibonacchi_Test.seedValue1(de.jdienst.Fibonacchi_Test) (0 ms)
- de.jdienst.Fibonacchi_Test.seedValue2(de.jdienst.Fibonacchi_Test) (0 ms)
- de.jdienst.Fibonacchi_Test.value11(de.jdienst.Fibonacchi_Test) (1 ms)

```
8   public class Sort
9   {
10
11    public static List<Integer> sort(List<Integer> coll)
12    {
13      List<Integer> list = new ArrayList<>();
14      list.addAll(coll);
15
16 1    Collections.sort(list);
17 1    log(list);
18
19 1    return Collections.unmodifiableList(list);
20    }
21
22    public static void log(List<Integer> list)
23    {
24 1    System.out.println(
25        list.stream().map(Object::toString)
26        .collect(Collectors.joining(", ")));
27    }
```

Mutations

16	1. removed call to java/util/Collections::sort → KILLED
17	1. removed call to de/jdienst/Sort::log → SURVIVED
19	1. mutated return of Object value for de/jdienst/Sort::sort to (if (x != null) null else throw new RuntimeException) → KILLED
24	1. removed call to java/io/PrintStream::println → SURVIVED

Active mutators

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

Tests examined

- de.jdienst.Sort_Test.twoList(de.jdienst.Sort_Test) (2 ms)
- de.jdienst.Sort_Test.emptyList(de.jdienst.Sort_Test) (51 ms)
- de.jdienst.Sort_Test.oneList(de.jdienst.Sort_Test) (3 ms)

Fazit

100% ≠ Fehlerfreiheit

Arbeitsaufwand

Bessere Testsuite

JohannesDienst

johannesdienst.net

info@johannesdienst.net