

# Code Coverage

---

von Sebastian Brehme

# Inhalt

Theorie

Arten von Code Coverage

Vor- und Nachteile

Tutorial Java Code Coverage in Eclipse

Tutorial Coverage im Continuous Integration Prozess

# Theorie

Testabdeckung

keine Angabe zur Qualität der Tests (dafür Mutationstests)

finden nicht getesteter Codestellen

meist keine 100% aus Kosten-Nutzen-Gründen

hohe Coverage durch schlechte Tests möglich

Anteil tatsächlicher durchgeführter Tests zu theoretisch möglichen

# Statement-Coverage

bei Test durchlaufenen Anweisungen : alle Anweisungen

Bsp: Wertzuweisung, Fallunterscheidung, Schleifen, Methodenaufruf

meist keine 100%  $\Rightarrow$  bei Fällen meist nicht jeder abgedeckt

schwacher Abdeckungsgrad

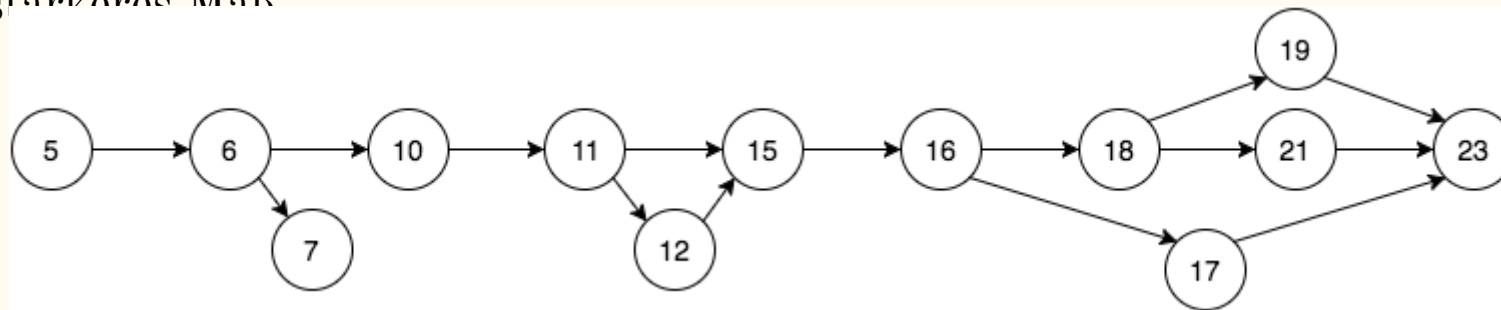
```
1 package coverage;
2
3 public class HealthChecker {
4     public static String getRecommendation(float size, float weight) {
5         boolean debug = true;
6         if (size > 240 || size < 100 || weight > 300 || weight < 30) {
7             return "Please enter size between 100 and 240 cm and weight between 30 and 300 kg";
8         }
9
10        float bmi = (weight * 10000) / (size * size);
11        if (debug) {
12            System.out.println("bmi: " + bmi);
13        }
14
15        String recommendation = "";
16        if (bmi < 20) {
17            recommendation = "Eat more";
18        } else if (bmi > 25) {
19            recommendation = "Eat less";
20        } else {
21            recommendation = "Don't change eating";
22        }
23        return recommendation;
24    }
25 }
```

# Branch-Coverage

Zweigabdeckung

Verhältnis der beim Test durchlaufenen Zweige

100% Zweig bedingt 100% Anweisungsabdeckung (nicht umgekehrt)  $\Rightarrow$   
stärkeres Maß



# Decision-Coverage

Bedingung mit Teilbedingungen  $\Rightarrow$  Werte jeder Teilbedingung

Einfachbedingungsabdeckung  $\Rightarrow$  jede Teilbedingung mindestens einmal wahr und einmal falsch

100% Coverage  $\neq$  100% Zweigabdeckung

Mehrfachbedingungsabdeckung  $\Rightarrow$  testen aller Kombinationen  $\Rightarrow$  sehr viele Testfälle notwendig  $\Rightarrow$  Sinn?

# Vorteile

ausreichende Tests  $\Rightarrow$  ausreichend fehlerfreie Software

notwendige Voraussetzung für ausreichende Tests  $\Rightarrow$  Code überhaupt durchlaufen

einfache Variante zum Erkennen der Vollständigkeit von Tests



# Nachteile

kein Korrektheitsbeweis der Software

nichte jedes Werkzeug unterstützt jede Variante  $\Rightarrow$  meist Statement
















Abdeckung von Bibliotheken meist nicht bestimmbar

hardwarenahe Anweisungen meist nicht vollständig testbar

# EclEmma Java Code Coverage

Tutorial:

1. Eclipse öffnen
2. Help ⇒ Eclipse Marketplace
3. “EclEmma” ⇒ Go
4. “EclEmma Java Code Coverage” installieren
5. Pfeil neben Symbol anklicken und JUnit Test auswählen
6. Ergebnis öffnet sich

BMITest (21.05.2017 22:27:13)					
Element	Coverage	Covered Instru...	Missed Instruct...	Total Instructio...	
▼ Coverage_E-Portfolio	 59,9 %	161	108	269	
▼ src	 59,9 %	161	108	269	
▼ main.java	 51,1 %	113	108	221	
▼ BMITest.java	 51,1 %	113	108	221	
▼ BMITest	 51,1 %	113	108	221	
• getResult(String, double)	 49,1 %	104	108	212	
• calculate(double, double)	100,0 %	6	0	6	
▼ test.java	 100,0 %	48	0	48	
▼ BMITestTest.java	 100,0 %	48	0	48	
▼ BMITestTest	 100,0 %	48	0	48	
• setUpBeforeClass()	 100,0 %	5	0	5	
• testCalculation()	 100,0 %	8	0	8	
• testResultInvalid()	 100,0 %	8	0	8	
• testResultMan()	 100,0 %	8	0	8	
• testResultWomen()	 100,0 %	8	0	8	
• testResultWrongNumbe	 100,0 %	8	0	8	

# Tutorial Coverage bei Codecov

Files	≡	●	●	●	Coverage
<a href="#">src/main/java/BMICalculator.java</a>	32	12	7	13	37.50%
<b>Project Totals</b> (1 files)	32	12	7	13	37.50%

✓ **master** Update pom.xml

-o- #2 passed

🔗 Commit cf87d0c

🕒 Ran for 54 sec

🔗 Compare 10a3064...cf87d0c

📅 25 minutes ago

🔗 Branch master

🔗 SebastianBrehme authored

🔗 GitHub committed

```
1142 ==> Uploading reports
1143 url: https://codecov.io
1144 query: branch=master&commit=cf87d0c00d0db45ddddd9f075bdcf765ec2515057&build=2.1&build_url=&name=&tag=&slug=SebastianBrehme%2FCoverage_E-Portfolio&yaml=&service=travis&flags=&pr=false&job=234891691
1145 -> Pinging Codecov
1146 -> Uploading to S3 https://codecov.s3.amazonaws.com
1147 -> View reports at https://codecov.io/github/SebastianBrehme/Coverage_E-Portfolio/commit/cf87d0c00d0db45ddddd9f075bdcf765ec2515057
1148
```

# Github Repository mit Projekt, Folien, Tutorials

[https://github.com/SebastianBrehme/Coverage\\_E-Portfolio](https://github.com/SebastianBrehme/Coverage_E-Portfolio)

# Quellen

<https://www.johner-institut.de/blog/iec-62304-medizinische-software/code-coverage/>

<https://martinfowler.com/bliki/TestCoverage.html>

<https://de.wikipedia.org/wiki/Testabdeckung>