Testgetriebene Entwicklung JUnit

Markus Lausberg
und
Robin Ziehe

Hochschule Aachen
Aachen University
of Applied Sciences

RWTH
Rechen- und
Kommunikationszentrum
RHEINISCH-
WESTFÄLISCHE
TECHNISCHE
HOCHSCHULE
AACHEN

Gliederung

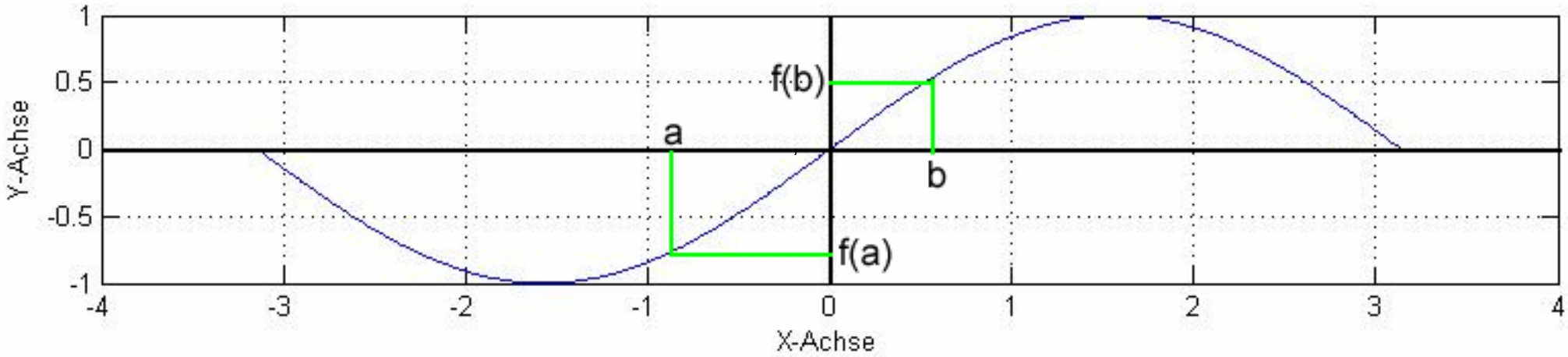
- Motivation
- Testgrundlagen
- Unit-Tests
- JUnit
- Bewertung
- Fragen

Motivation

- bürokratischen Aufwand reduzieren
- Zustand von Software in Entwicklung?
- Debugging teuer
- Erweiterbarkeit
- Anforderungen beschreiben
- Testaufwand minimieren

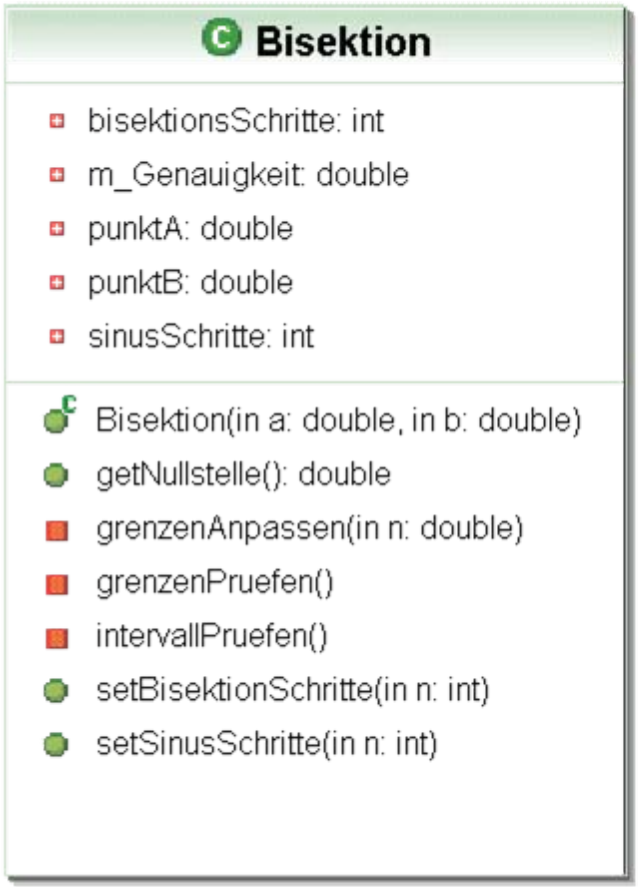
Nullstellenberechnung der Sinusfunktion mit dem Bisektionsverfahren

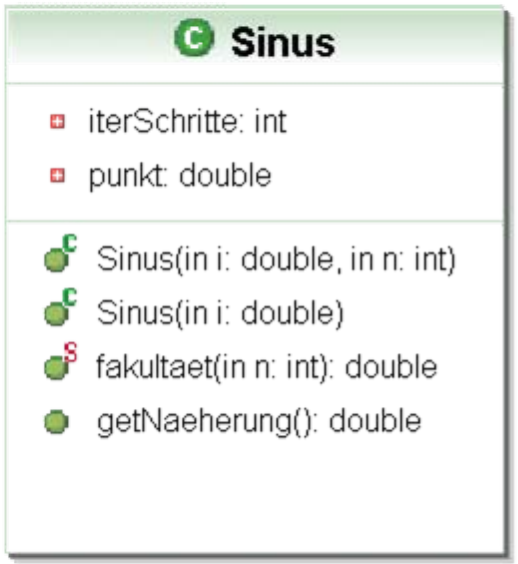
$$\sin(x) = \sum_{i=0}^{n} (-1)^i \frac{x^{(2i+1)}}{(2i+1)!}, \; n \in \Re^+$$



-falsche Eingabewerte
-Genauigkeit

Klassenaufbau





Testgrundlagen

- White-Box-Test
  - Codeabdeckung
- Black-Box-Test
  - Grenz- Extremwerte
  - Äquivalenzklassen
- Grey-Box-Test

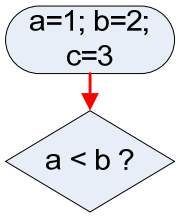
White-Box-Test

- Quellcode bekannt
- Test gegen die Implementierung
- Codeabdeckung
  - Anweisungsüberdeckung
  - Zweigüberdeckung
  - Pfadüberdeckung

Codeabdeckung (1)

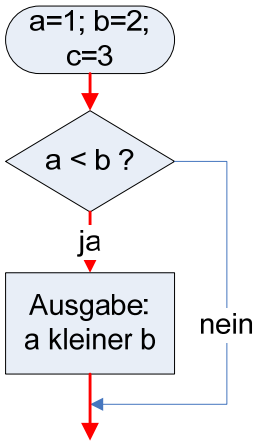
```
1  public class Beispiel {
2      public void ueberdeckung(int a, int b, int c) {
3          if (a < b) {
4              System.out.println("a ist kleiner als b");
5          }
6          if (b < c) {
7              System.out.println("b ist kleiner als c");
8          }
9      }
10 }
```

Codeabdeckung (2)



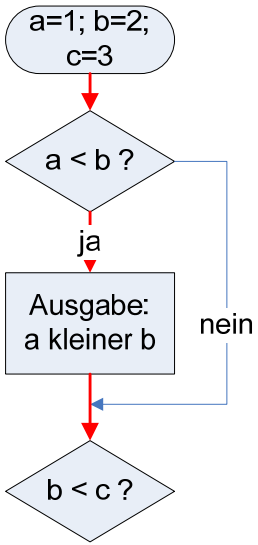
Anweisungsüberdeckung

Codeabdeckung (2)



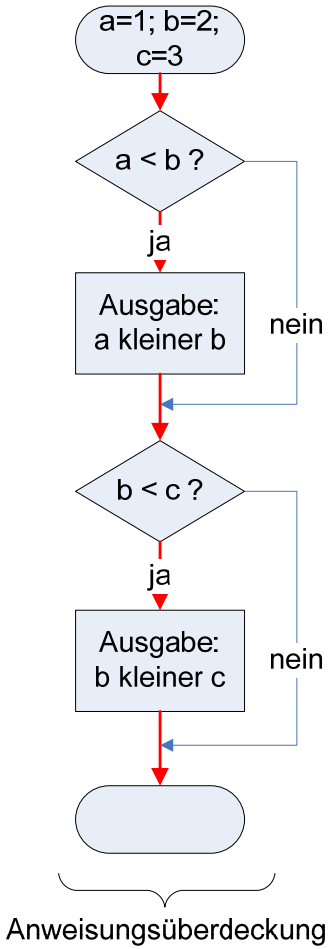
Anweisungsüberdeckung

Codeabdeckung (2)

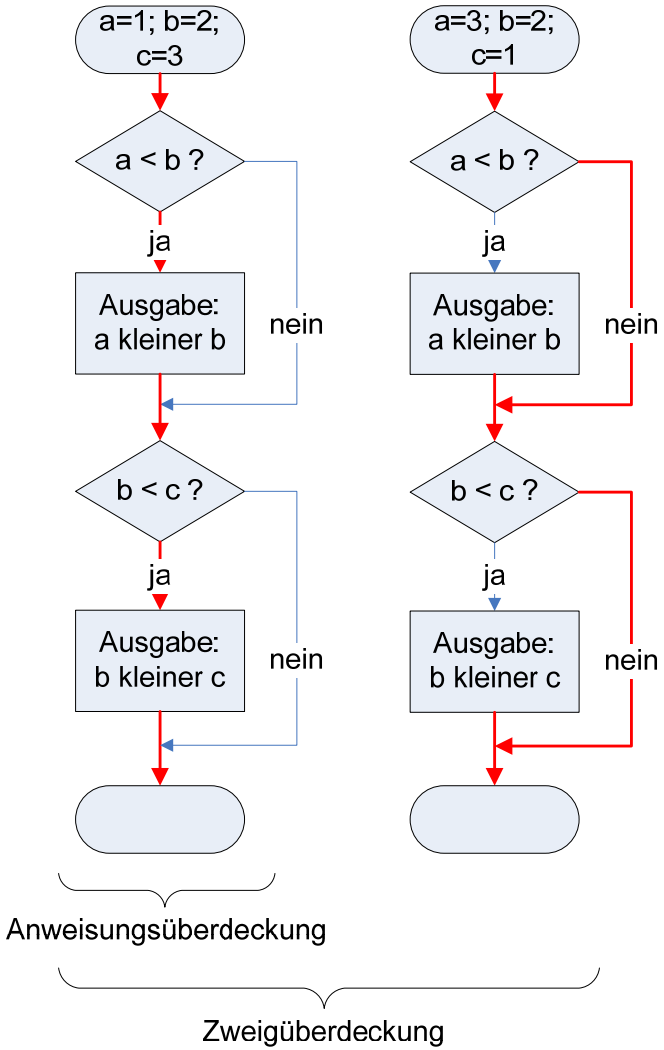


Anweisungsüberdeckung

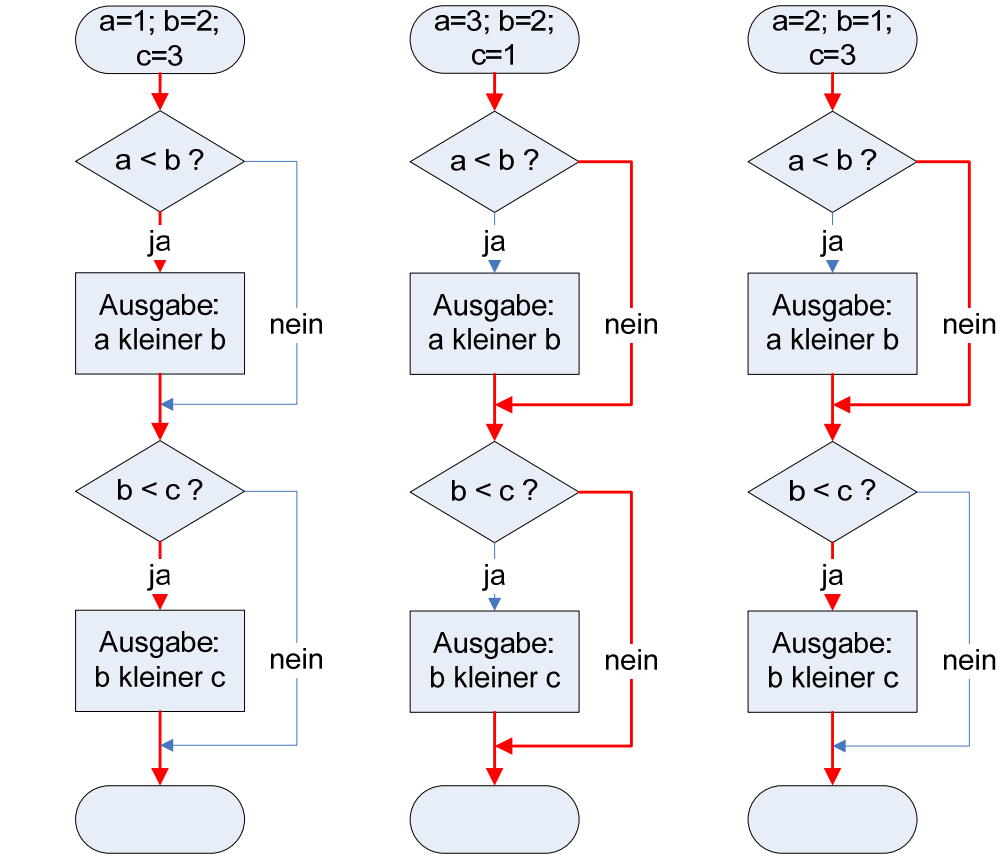
Codeabdeckung (2)



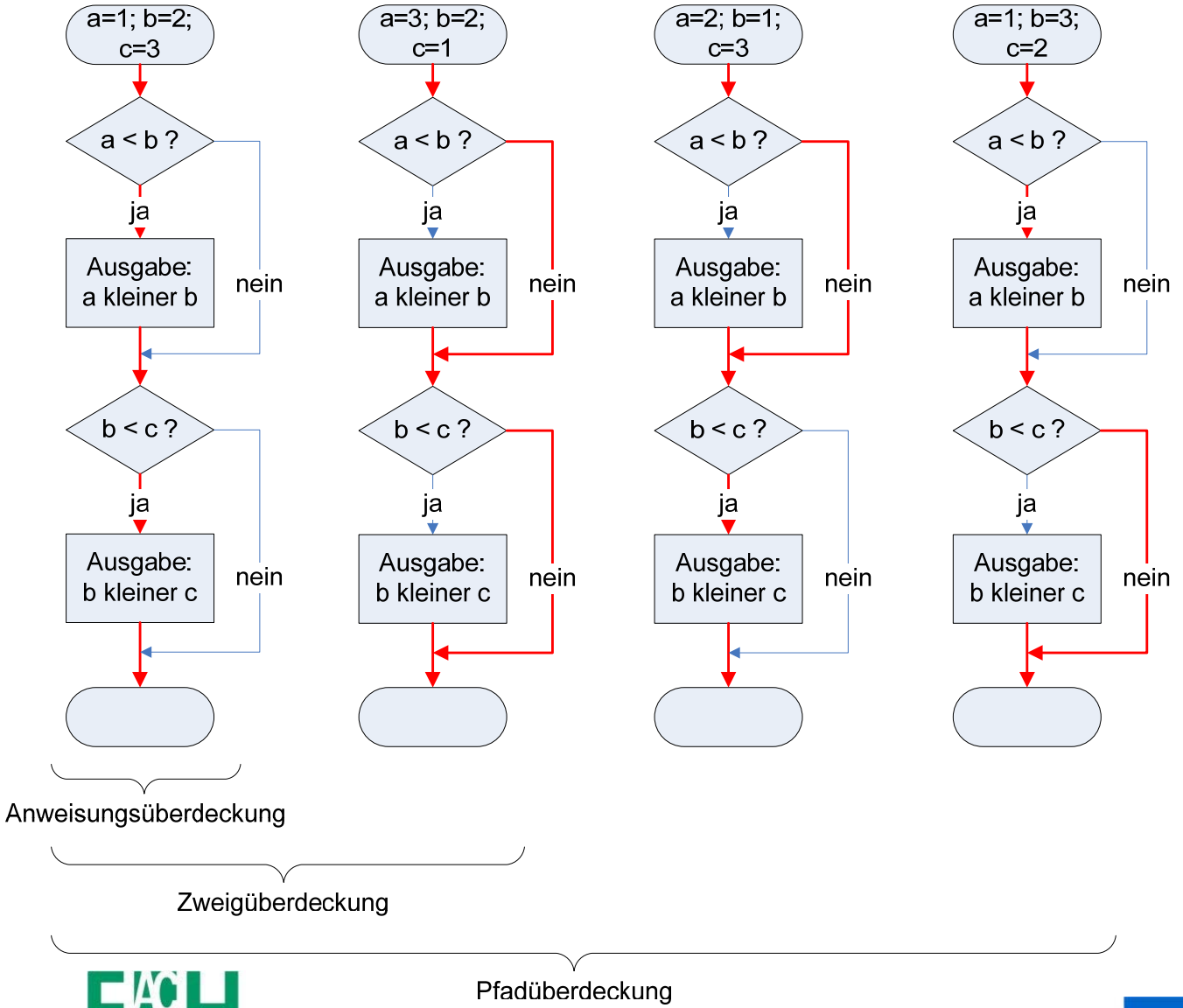
Codeabdeckung (2)



Codeabdeckung (2)



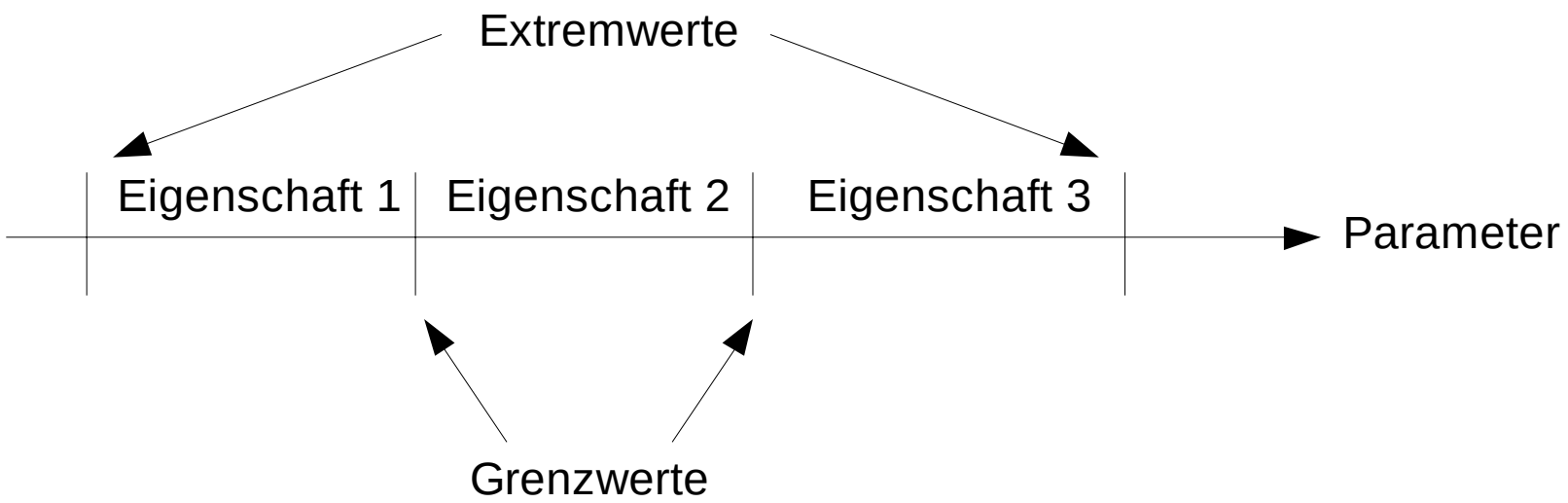
Codeabdeckung (2)



Black-Box-Test

- Quellcode unbekannt
- Test gegen die Spezifikation
- Grenz- und Extremwerte
- Äquivalenzklassen

Grenz- und Extremwerte



Grey-Box-Test

- Vorteile von Black- und White-Box-Tests
- Zuerst Black-Box
- Dann White-Box

Unit-Tests

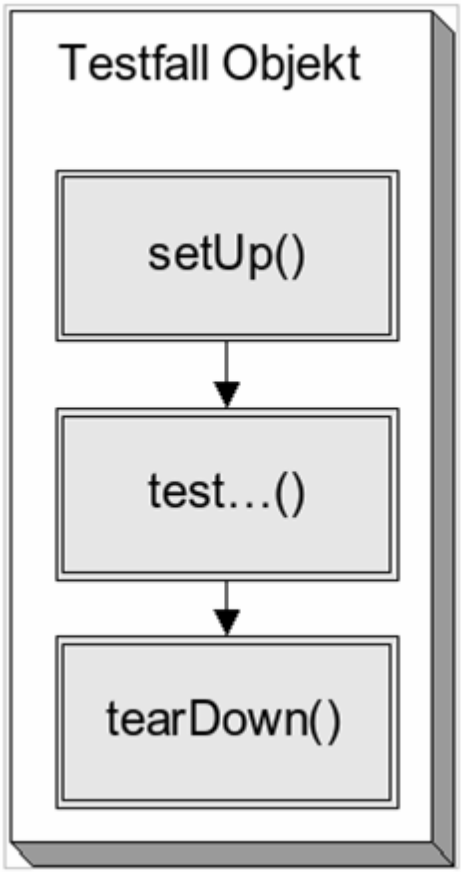
- Isolierte Tests
  - Framework
  - Getestet werden
    - Methoden
    - Module
    - Klassen
- Testgetriebenes Programmieren
  1. Test schreiben
  2. Klasse schreiben
  3. Klasse testen
- Testplan

JUnit, als Testrahmenwerk

- www.junit.org, JUnit Version 3.8
- Framework
  - TestCase
    - Fixture
    - Assert-Methoden
    - Exception Handling in JUnit
  - TestSuite

TestCase

- Fixture
  - setUp()
    - Initialisieren
  - test…()
    - Methodentest
  - tearDown()
    - Löschen



TestCase

- Assert-Methoden
  - Variablen und Ergebnis Kontrolle
  - Initialisierung prüfen
  - Boolsche Ausdrücke verwerten

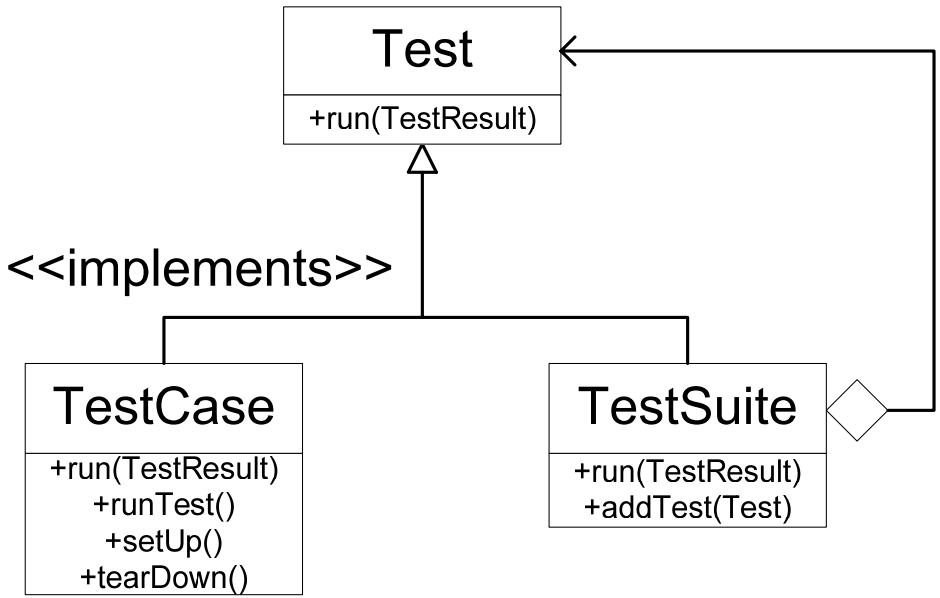
TestCase

- Exception Handling in JUnit
  - Failure
    - Assert-Methode fehlgeschlagen
  - Error
    - Unerwartete Exception aufgetreten

TestSuite

TestCase und TestSuite zu einer TestSuite zusammenfassen

Implementierung in JUnit



Bewertung

- Gute Testfälle können durch nichts ersetzt werden
- Bestehenden Code zu migrieren ist schwierig

Fragen