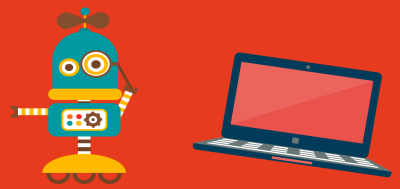
htl3 rennweg

Test Driven Development & JUnit

DI Reinhold Buchinger

Höhere Abteilung für Mechatronik
Höhere Abteilung für Informationstechnologie
Fachschule für Informationstechnik



Ziele

- Das Konzept von Test Driven Development verstehen und anwenden können
- JUnit Tests für eigene Programme entwickeln

JUnit

Tests bisher

- Ausgaben machen

```
System.out.println("Ergebnis: " + MeineKlasse.meineMethode(""));
System.out.println("Ergebnis: " + meinObjekt.meineMethode(""));
```

- Zahlreiche Nachteile
  - Schwer zu nachvollziehen ob Ausgaben korrekt sind.
  - Schwer zu verwalten
  - Tests und Programmcode nicht getrennt

JUnit

- Framework zum Testen von Java Programmen
- JUnit ist nicht Teil vom (Standard-) Java sondern muss extra hinzugefügt werden.
- Wir verwenden JUnit5 (nicht JUnit4)

Testklasse

- Eine Gruppe von Tests wird in einer Klasse zusammengefasst.
- Bei "Unit Tests" wird immer eine "Unit" isoliert getestet
  - Bei uns ist eine "Unit" eine Klasse
  - Eine Testklasse für jede Klasse
- Namenskonvention: Klasse "Person" -> Testklasse "PersonTest" (ein "Test" an den Klassennamen anhängen)
- Testklasse automatisch von IntelliJ generieren lassen
- Code von Testcode trennen (zumindest verschiedene packages)

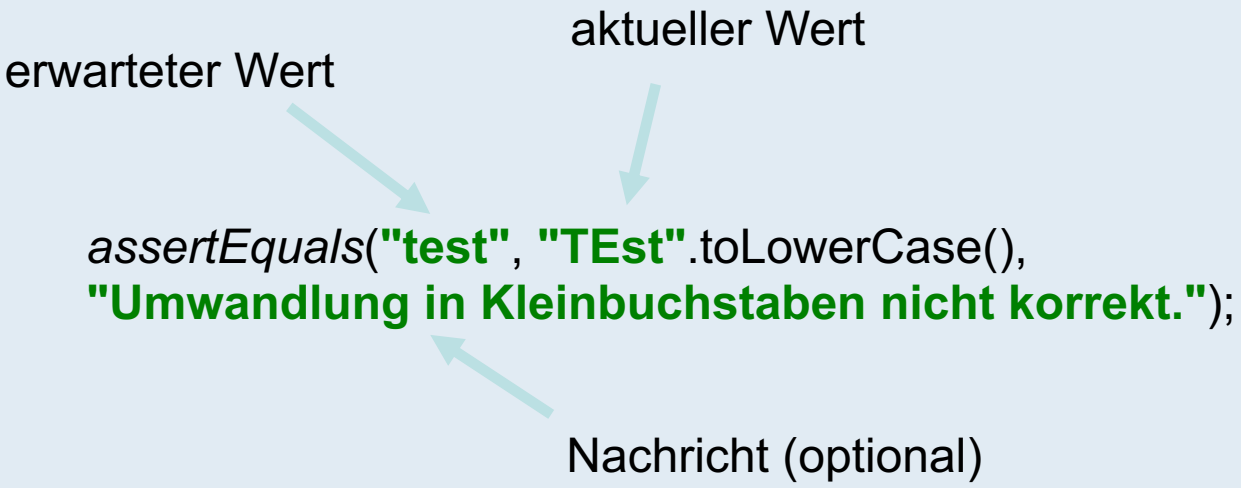
Testmethode


```
@Test
public void getFirstName_single(){

}
```

Assertions

- Es stehen verschiedene Assertions zu Verfügung
  - https://junit.org/junit5/docs/current/api/org.junit.jupiter.api/org/junit/jupiter/api/Assertions.html
- Ist die Bedingung nicht erfüllt, bricht der Test ab und gibt eine entsprechende Fehlermeldung aus.
- Beispiel:



@BeforeAll, @BeforeEach,...

htl3 rennweg
IT & MECHATRONIK

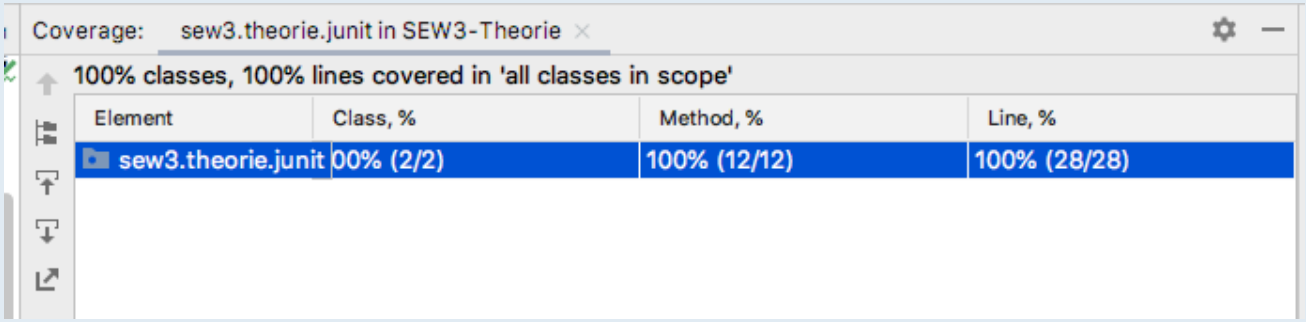
- Methoden können mit weiteren Annotations ausgezeichnet werden.
  - @BeforeAll: Einmal vor allen Tests der Testklasse ausgeführt
  - @BeforeEach: Vor jedem einzelnen Test ausgeführt
  - @AfterAll: Nach allen Tests der Testklasse ausgeführt
  - @AfterEach: Nach jedem einzelnen Test ausgeführt

Grenzfälle

- Immer Grenz-/Sonderfälle testen ("Corner Cases")!
  - leere Texte.
  - `null` Werte.
  - Liste: leer, ein Element, viele Elemente.
  - Werte **die in einem `if/while` vorkommen**.
  - Wenn x ein Sonderfall ist: x-1, x und x+1 ebenfalls testen.

Code Coverage

- Code Coverage (Testabdeckung) bezeichnet das Verhältnis von Code, der durch Tests ausgeführt wird, zum gesamten Code.
- Code Coverage Report von IntelliJ erstellen lassen.
- In den Übungen wollen wir 100% Code Coverage.
  - In der Praxis aufgrund von Kosten-Nutzen meist eine etwas geringere Abdeckung

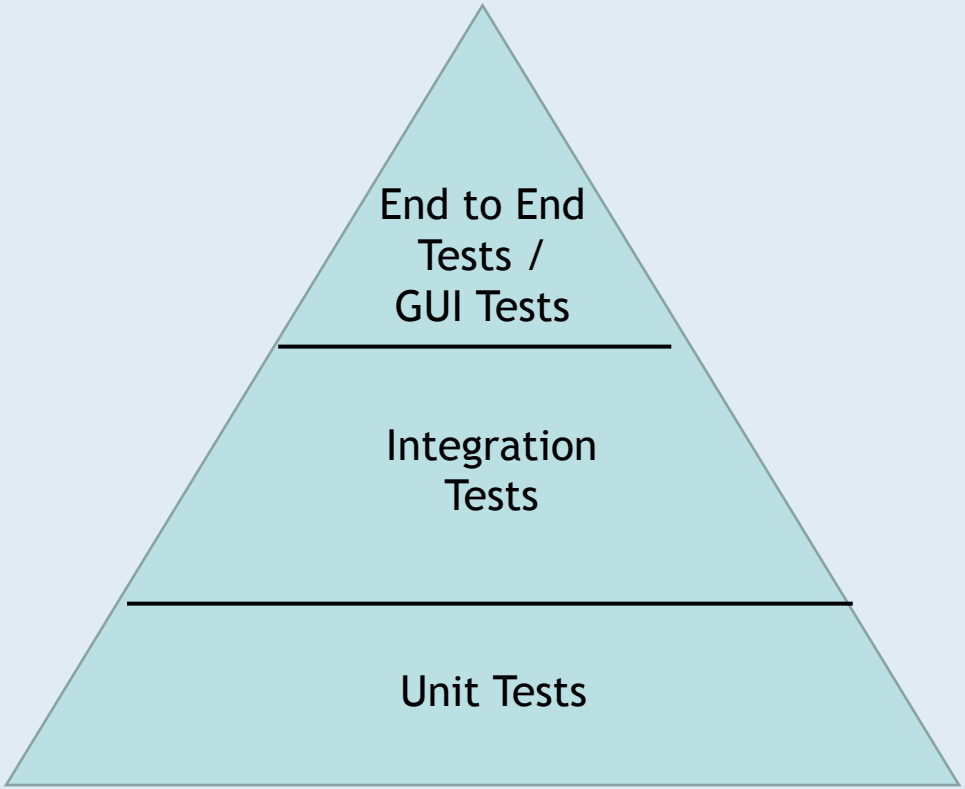


Testhierarchie

wenige Tests

aufwändig

langsam



viele Tests

einfach

schnell

Testet das Gesamtsystem unter realen Bedingungen

Testet die Kombination mehrerer Units

Testet isoliert einzelne Units (häufig Klassen)

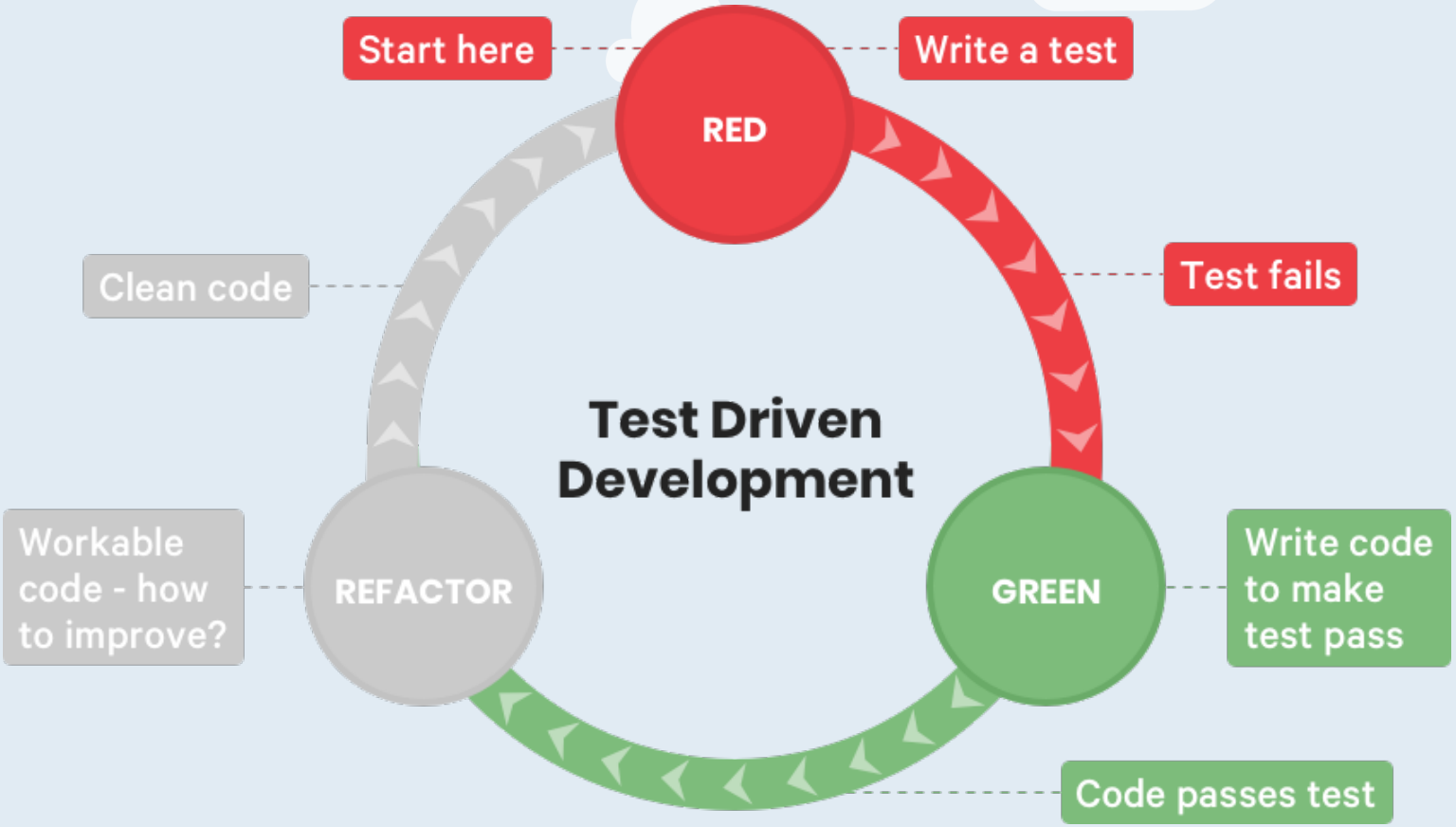
Test Driven Development (TDD)

Test Driven Development (TDD)

htl3 rennweg
IT & MECHATRONIK

- Bestimmte Vorgehensweise beim Programmieren (Entwicklungsmethode)
- Tests werden **vor** dem Code geschrieben.

Test Driven Development -



Fragen?
Anregungen?
Bemerkungen?