

Java Workshop
Einführung in die Testgetriebene Entwicklung mit JUnit
JUnit Annotations, TestSuites, ExecutionOrder

Thomas Staubitz
Ralf Teusner

- In welcher Reihenfolge werden die Tests ausgeführt?
 - Erwartung: Reihenfolge im Code

- Ist die Reihenfolge in der die Tests ausgeführt werden überhaupt wichtig?
 - Ganz im Gegenteil.
 - Man sollte sich keineswegs darauf verlassen, dass Tests in einer bestimmten Reihenfolge angegangen werden.
 - Tests sollten grundsätzlich autark sein und nicht aufeinander aufbauen.
 - <https://github.com/junit-team/junit4/wiki/Test-execution-order>

Java - JUnit

Thomas Staubitz,
Ralf Teusner

- Anders ist es bei den Fixtures, hier ist die Reihenfolge klar vorgegeben
 - @BeforeClass
 - @Before
 - ...
 - @After
 - @AfterClass

Java Annotations

- Java Annotations kennen wir schon aus anderen Zusammenhängen
 - z.B. @Override
- Java Annotations
 - sind Meta-Daten über das Programm
 - liefern Informationen
 - für den Compiler
 - Software Tools
 - Manche Annotations können auch zur Laufzeit ausgewertet werden

Java - JUnit

Thomas Staubitz,
Ralf Teusner

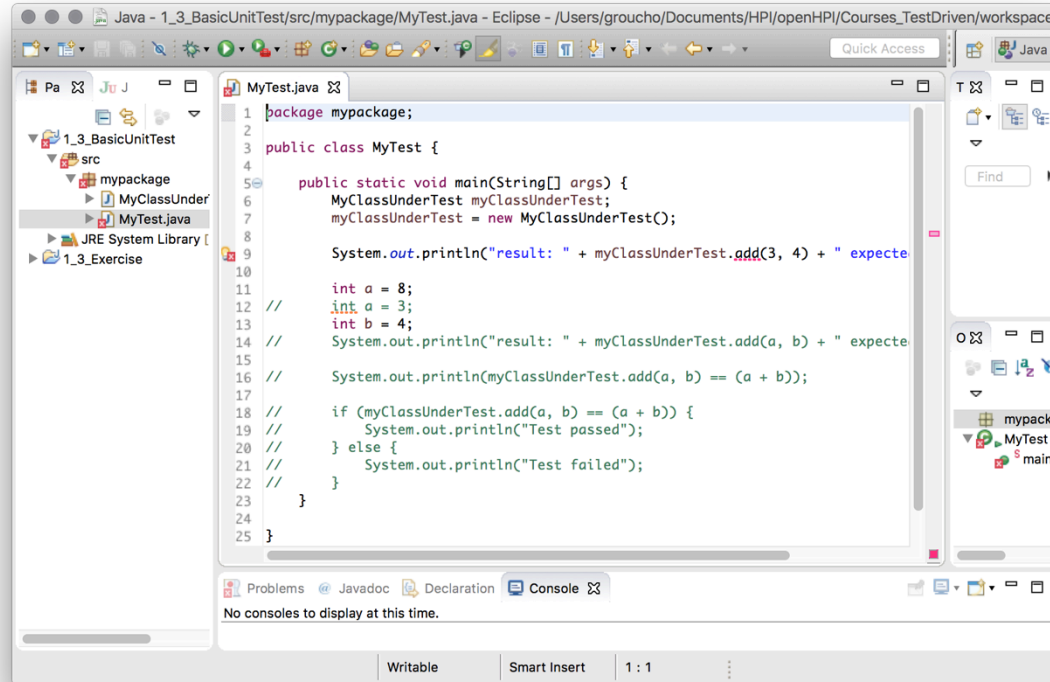
Annotations im JUnit Kontext

- @Test – haben wir schon vorgestellt, es hilft z.B. der IDE herauszufinden, dass die folgende Methode ein Test ist und entsprechende Menuelemente bereitzustellen
- @Before – Methoden die mit @Before annotiert sind werden einmal **vor** jedem @Test ausgeführt
- @After – Methoden die mit @After annotiert sind werden einmal **nach** jedem @Test ausgeführt
- @BeforeClass – Methoden die mit @BeforeClass annotiert sind werden **einmal vor** allen Tests der Testklasse ausgeführt
- @AfterClass – Methoden die mit @AfterClass annotiert sind werden **einmal nach** allen Tests der Testklasse ausgeführt

Java - JUnit

Thomas Staubitz,
Ralf Teusner

Annotations im JUnit Kontext



```
1 package mypackage;
2
3 public class MyTest {
4
5     public static void main(String[] args) {
6         MyClassUnderTest myClassUnderTest;
7         myClassUnderTest = new MyClassUnderTest();
8
9         System.out.println("result: " + myClassUnderTest.add(3, 4) + " expected");
10
11         int a = 8;
12         // int a = 3;
13         int b = 4;
14         // System.out.println("result: " + myClassUnderTest.add(a, b) + " expected");
15
16         // System.out.println(myClassUnderTest.add(a, b) == (a + b));
17
18         // if (myClassUnderTest.add(a, b) == (a + b)) {
19         //     System.out.println("Test passed");
20         // } else {
21         //     System.out.println("Test failed");
22         // }
23     }
24 }
25 }
```

Java - JUnit

Thomas Staubitz,
Ralf Teusner

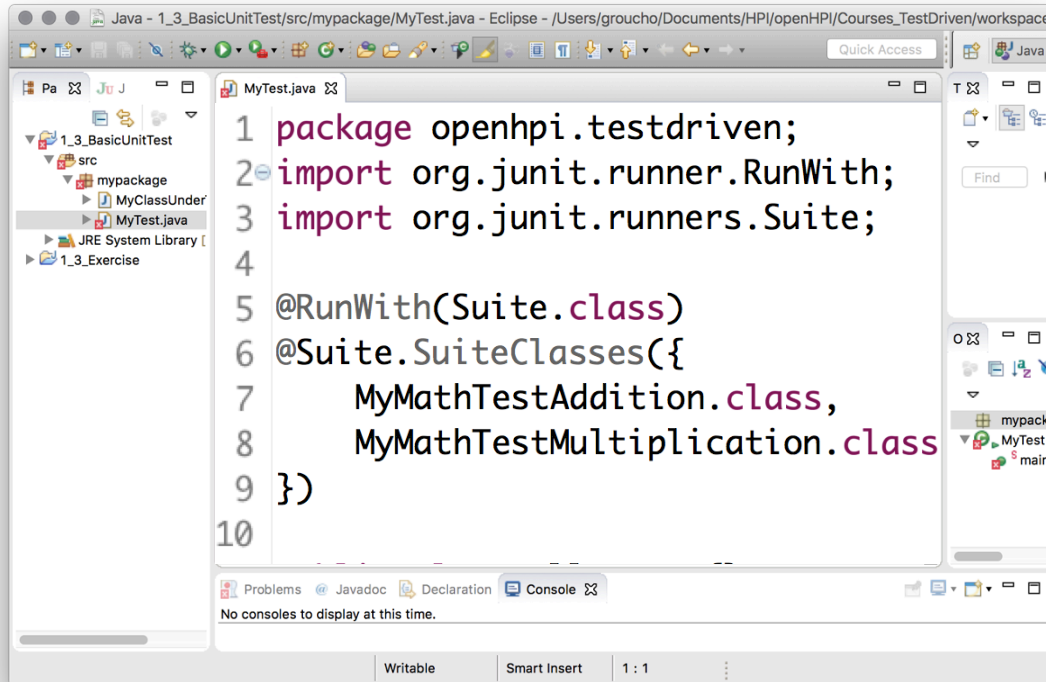
Testsuites

- In Testsuiten werden mehrere Testklassen in bestimmten Abläufen zusammengefasst.
- Dadurch muss ich nicht mehr alle Tests einzeln aufrufen sondern kann mehrere Tests automatisiert nach einander laufen lassen.
- Testsuiten werden ebenfalls mittels Annotations realisiert
 - @RunWith(Suite.class)
 - @Suite.SuiteClasses
- Bei größeren Projekten wird man in der Regel noch einen Schritt weiter gehen und ein Tool wie ANT, Maven oder Gradle bzw. einen CI Server wie z.B. Jenkins, Hudson, TeamCity oder TravisCI einsetzen.

Java - JUnit

Thomas Staubitz,
Ralf Teusner

Testsuite Beispiel



```
1 package openhpi.testdriven;
2 import org.junit.runner.RunWith;
3 import org.junit.runners.Suite;
4
5 @RunWith(Suite.class)
6 @Suite.SuiteClasses({
7     MyMathTestAddition.class,
8     MyMathTestMultiplication.class
9 })
10
```

Java - JUnit

Thomas Staubitz,
Ralf Teusner



Java Workshop

Einführung in die Testgetriebene Entwicklung mit JUnit

JUnit Annotations, TestSuites, ExecutionOrder

Thomas Staubitz
Ralf Teusner