SEW3

IT-Medientechnik

Unittests mit JUnit 5

https://junit.org/junit5/docs/current/user-guide/

TWS

Abhängigkeiten in pom.xml eintragen

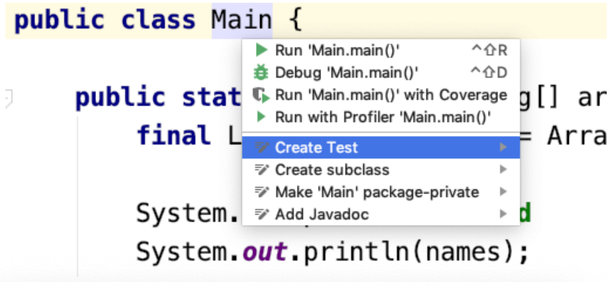
```xml
<properties>
    <maven.compiler.source>11</maven.compiler.source>
    <maven.compiler.target>11</maven.compiler.target>
    <junit.jupiter.version>5.5.1</junit.jupiter.version>
</properties>

<dependencies>
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter</artifactId>
        <version>${junit.jupiter.version}</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.hamcrest</groupId>
        <artifactId>hamcrest-all</artifactId>
        <version>1.3</version>
        <scope>test</scope>
    </dependency>
</dependencies>
```

Tests generieren

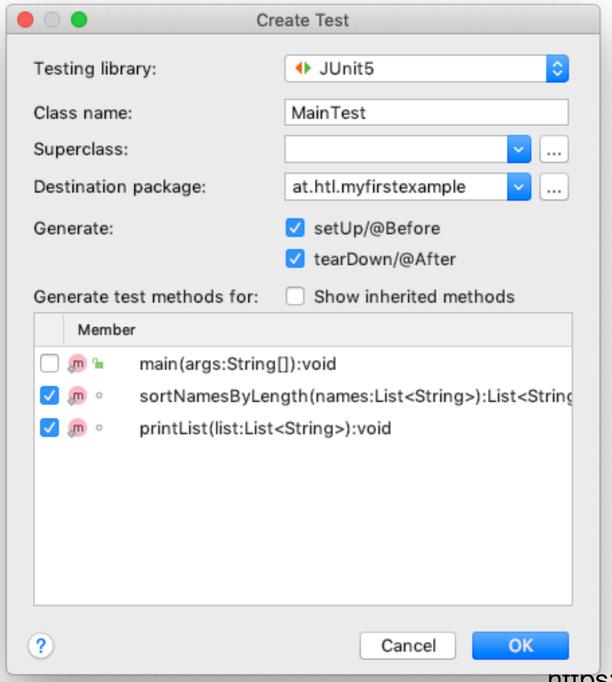
- 1.Cursor (Caret) auf den Klassennamen setzen

```
public class Main {
```

- 2.<Alt> <Enter> drücken



- 3.Create Test

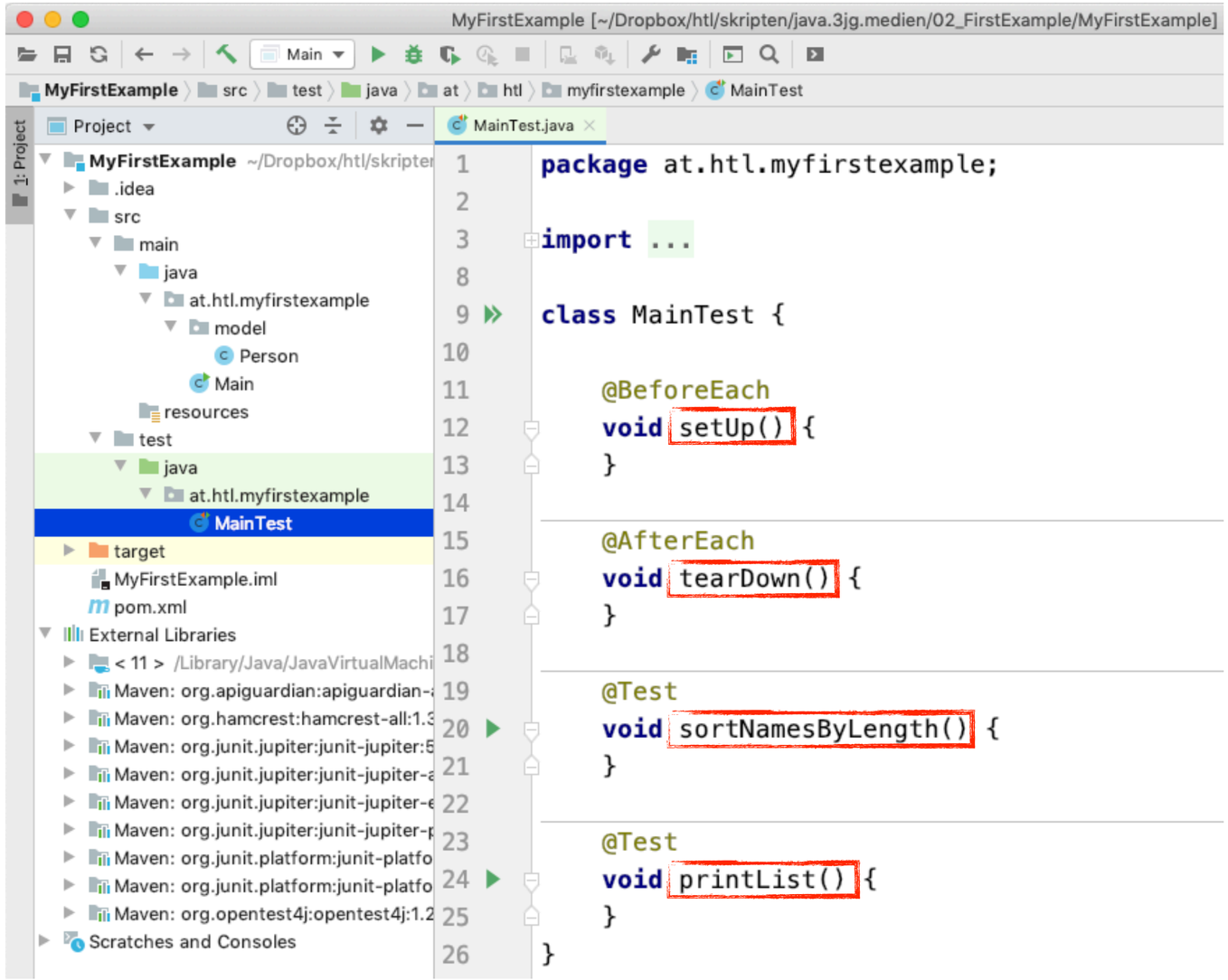


- 4.Ok

Verwendet man Maven ist die Directory-Struktur definiert (Maven Standard Directory Layout). Man muss kein Root-Testverzeichnis oder sonstiges definieren

https://maven.apache.org/guides/introduction/introduction-to-the-standard-directory-layout.html

Testgerüst wird generiert

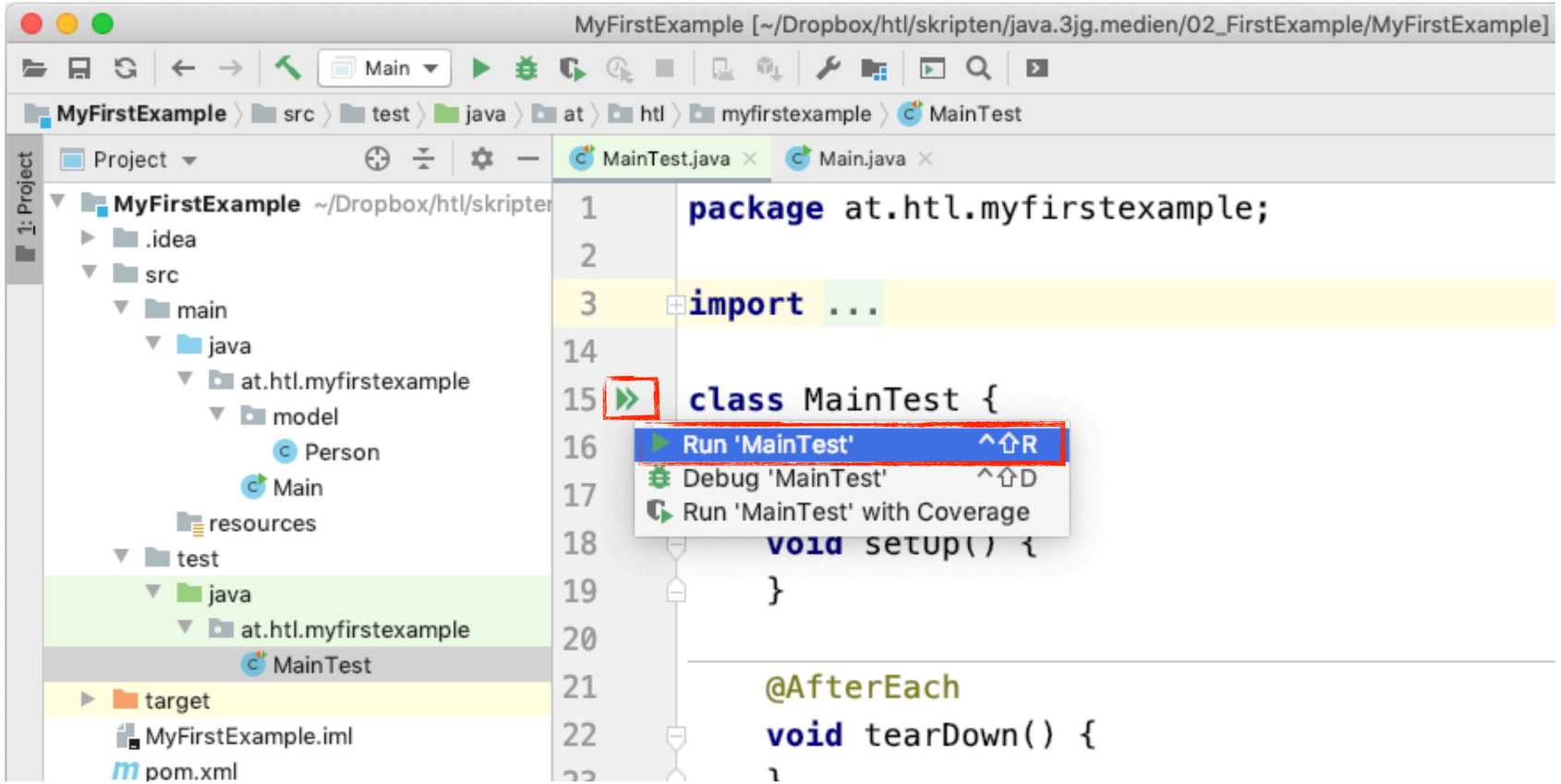


JUnit-Annotations

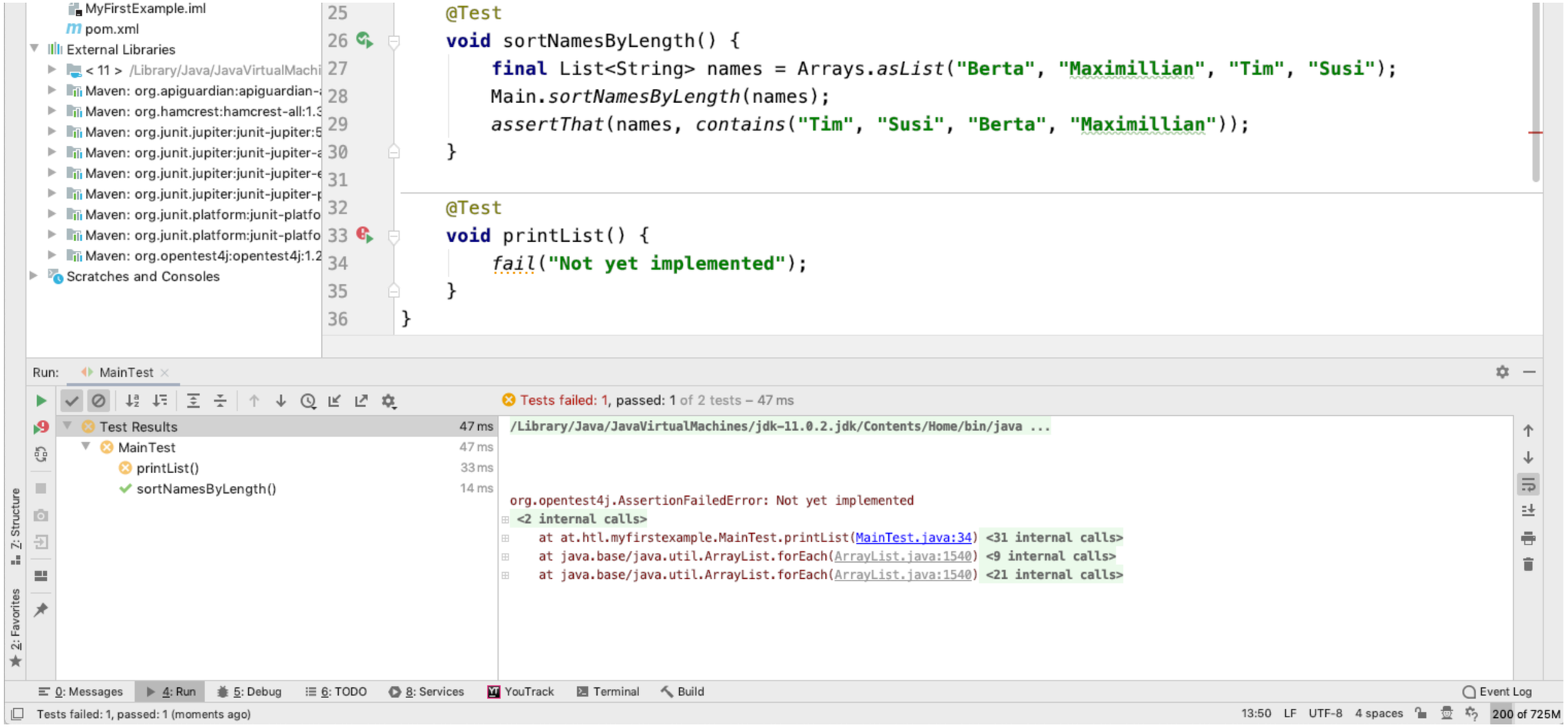
`@BeforeEach`	Denotes that the annotated method should be executed *before* **each** `@Test`, `@RepeatedTest`, `@ParameterizedTest`, or `@TestFactory` method in the current class; analogous to JUnit 4's `@Before`. Such methods are *inherited* unless they are *overridden*.
`@AfterEach`	Denotes that the annotated method should be executed *after* **each** `@Test`, `@RepeatedTest`, `@ParameterizedTest`, or `@TestFactory` method in the current class; analogous to JUnit 4's `@After`. Such methods are *inherited* unless they are *overridden*.
`@BeforeAll`	Denotes that the annotated method should be executed *before* **all** `@Test`, `@RepeatedTest`, `@ParameterizedTest`, and `@TestFactory` methods in the current class; analogous to JUnit 4's `@BeforeClass`. Such methods are *inherited* (unless they are *hidden* or *overridden*) and must be `static` (unless the "per-class" test instance lifecycle is used).
`@AfterAll`	Denotes that the annotated method should be executed *after* **all** `@Test`, `@RepeatedTest`, `@ParameterizedTest`, and `@TestFactory` methods in the current class; analogous to JUnit 4's `@AfterClass`. Such methods are *inherited* (unless they are *hidden* or *overridden*) and must be `static` (unless the "per-class" test instance lifecycle is used).

https://junit.org/junit5/docs/current/user-guide/#writing-tests-annotations

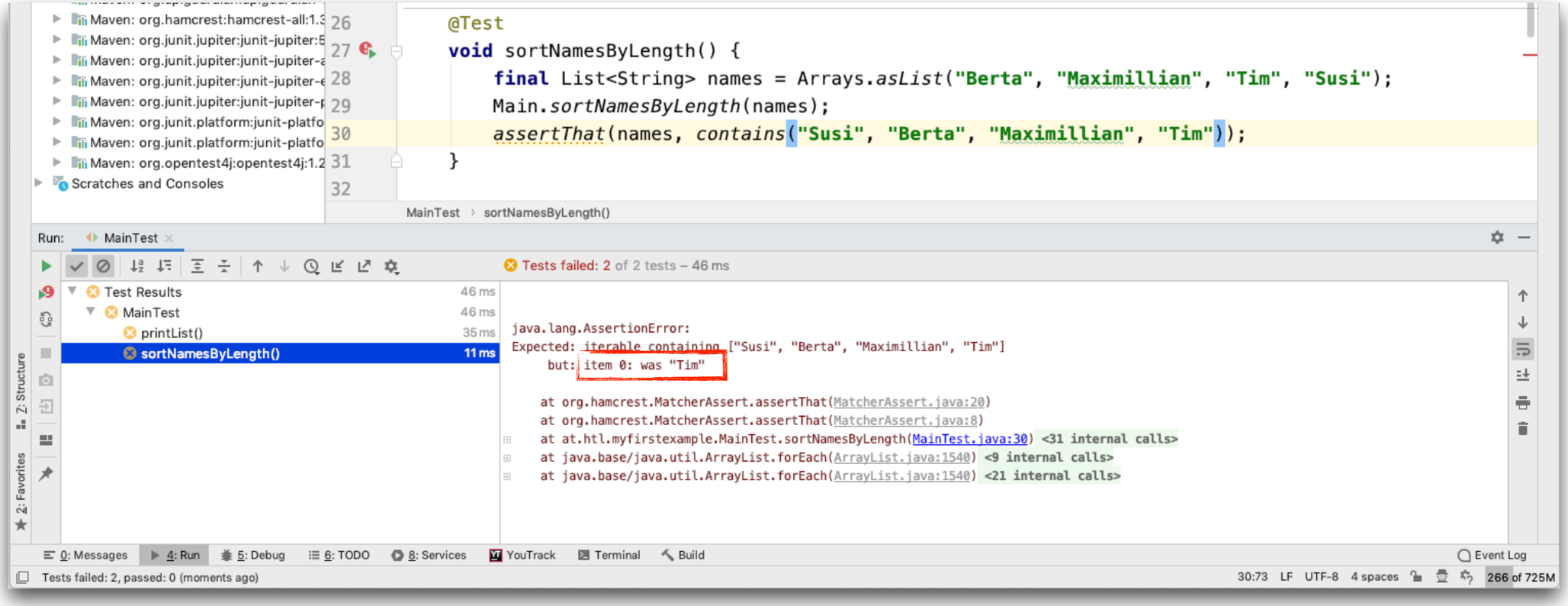
Starten des Testlaufs



Testergebnis



Tests fehlschlagen lassen

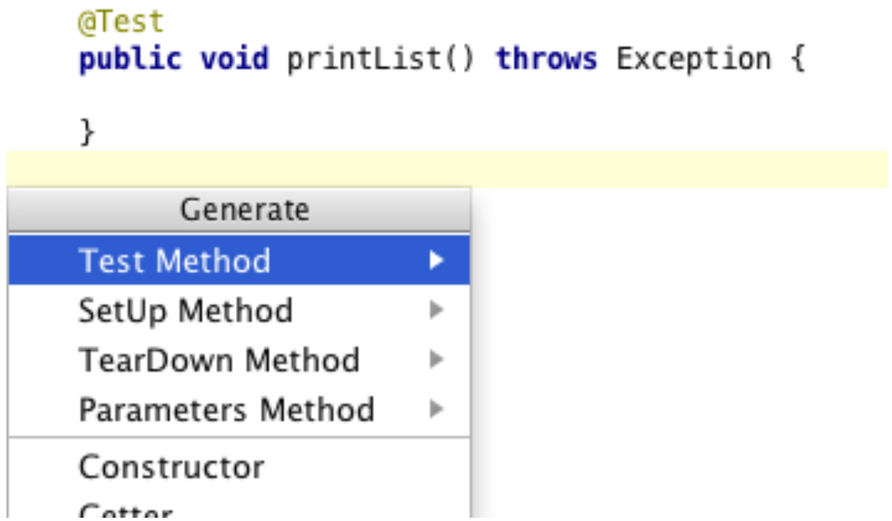


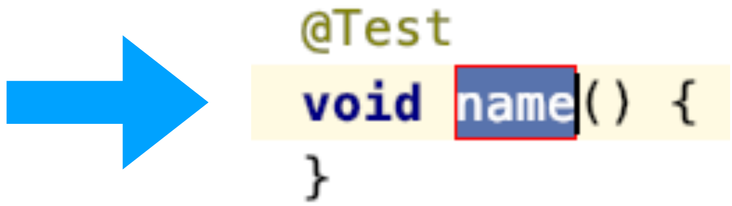
Während des Entwickelns soll man die einzelnen Tests auch einmal fehlschlagen lassen, um Ihre Funktionsfähigkeit zu überprüfen

Anlegen einer neuen Testmethode

Cursor in leere Zeile innerhalb der Klasse stellen:

⌘N + ↵





Reihenfolge der Testdurchführung

```java
import org.junit.jupiter.api.MethodOrderer.Alphanumeric;

@TestMethodOrder(Alphanumeric.class)
class MainTest {
```

JUnit Jupiter provides the following built-in `MethodOrderer` implementations.

- `MethodOrderer.Alphanumeric`
- `MethodOrderer.OrderAnnotation`
- `MethodOrderer.Random`

https://blog.codeleak.pl/2019/03/test-execution-order-in-junit-5.html

Bibliotheken für Tests

- SystemRules (leider derzeit nur für jUnit4)
  https://stefanbirkner.github.io/system-rules/
- Java-Faker
  https://github.com/DiUS/java-faker
- Log Collectors
  https://github.com/haasted/TestLogCollectors
- Awaitility (zum Testen asynchroner Methodenaufrufe - wartet bis eine Bedingung erreicht ist)
  https://github.com/awaitility/awaitility
- EqualsVerifier
  https://jqno.nl/equalsverifier/
- Make-it-easy
  https://github.com/npryce/make-it-easy

HTL LEONDING

Noch
Fragen?

HTL LEONDING

Schön, hier zu lernen