JUnit 5

1) Was sind Unit-Test:

Unit-Tests sind dazu da um verschiedene Funktionen und Teile eines Codes zu testen. Sie werden verwendet um verschiedene Fehler und Bugs zu finden, und sie danach zu lösen. Durch Unitest wird gezielt die Qualität des Projekt gefördert. Mehr Unitests → größere Qualität.

---

2 & 3) Wie erstellt man Unittests in JUnit5

- Wesentliches:
  - dependecy in pom.xml hinzufügen

```
<dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-engine</artifactId>
    <version>5.8.1</version>
    <scope>test</scope>
</dependency>
```

- Location der Tests und Aufbau:
  - In einem eigenen Source folder:
  - src/main/java - for Java classes
  - src/test/java - for test classes
  - Es wird eine eigene Test Class erstellt, wo man die Unit Tests erstellt
  - Diese Unit Tests sind Methoden, welche sich in der Test Class befinden

---

4) Wichtigsten Annotationen

@Test

- @Test Annotation markiert die Methode als Test - Methode

@ParameterizedTest

- Funktioniert wie die @Test Annotation, jedoch können Test - Methoden mehrmals mit unterschiedlichen Argumenten ausgeführt werden.

@RepeatedTest

- Wiederholt den Test. Man kann die Anzahl wie oft der Test wiederholt werden soll angeben

@DisplayName

- Gibt dem Test einen Namen, welcher dann bei zum Beispiel einem Report sichtbar ist

@BeforeEach

- Methoden mit dieser Annotation werden vor jeder Test - Methode ausgeführt

@AfterEach

@AfterEach

- Methoden mit dieser Annotation werden nach jeder Test - Methode ausgeführt

@BeforeAll

- Methoden mit dieser Annotation werden vor allen Test - Methoden ausgeführt

@AfterAll

- Methoden mit dieser Annotation werden erst ausgeführt, wenn alle Test - Methoden beendet wurden

@Disabled

- Diese Annotation disabled oder überspringt Tests auf Klassen oder Methoden Ebene.
  - Klassen Ebene → Jede Methode in der Klasse wird disabled
  - Methoden Ebene → Nur die Methoden mit der @Disabled Annotation werden disabled

---

5) Assertions (evtl. Assumptions)

assertion = Behauptung

Definition bzw. Erklärung

Assert ist eine Methode zur **Bestimmung des Pass- oder Fail-Status** eines Testfalls. Die Assert-Methoden werden von der Klasse **org.junit.Assert** bereitgestellt, die die Klasse **java.lang.Object** erweitert.

Junit bietet eine **Klasse namens Assert**, die eine Reihe von **Assertion-Methoden** bereitstellt, die beim **Schreiben von Testfällen** und bei der **Erkennung von Testfehlern** nützlich sind.

Die Assert-Methoden werden von der Klasse org.junit.Assert bereitgestellt, die die Klasse java.lang.Object erweitert.

---

6) Dynamische Tests

Es gibt normale unitest mit @Test Annotation sie werden erstellt ,wenn man das Projekt kompiliert.

```
Es gibt Dynamische Tests sie werden in runtime erstellt (erst wenn man es braucht)
mit der Annotation @TestFactory.
```

---

7) Parameterized Tests

Parameterized Tests sind normale unitest nur mit parametern

---

8) Testreihenfolge festlegen

Normal sollte die Testreihenfolge ja nicht festgelegt werden. Da eine Test-Methode nicht von einer anderen abhängig sein sollte.

- Es wird auch **für die Klasse** noch die **@TestMethodOrder(OrderAnnotation.class) Annotation** benötigt.

- Mit der @Order Annnotation und einen **Integer Parameter** wird die **Reihenfolge festgelegt**.
- Gibt es keine **@Order Annotation** an eine Methode werden erst die mit **@Order definierte Reihenfolge** verwendet und die ohne dann hinten von oben angefügt (= @Order(Integer.MAX_VALUE))

Es muss für die Ausführung auch der JUnit 5 Runner eingestellt werden, sonst läuft es nicht.

Last updated 2022-10-14 14:43:01 +0200