

Testen

SS 2019

Unit-Tests

- ▶ zu jeder Klasse, die getestet werden soll, wird eine Testklasse erstellt
- ▶ In der Testklasse werden
 - ▶ Instanzen der Klasse erstellt, die man testet
 - ▶ ein paar Methoden dieser Klasse aufgerufen und
 - ▶ über sogenannte Assertions überprüft, ob die Objekte auch das machen, was sie sollen.
- ▶ Automatisiertes Ablaufen dieser Klasse in Testframework
 - ▶ Z. B. JUnit

JUnit*

- ▶ als Bibliothek dem Classpath des Projekts hinzufügen
- ▶ Rechtsklick auf Projekt in Eclipse:
 - ▶ nacheinander Properties -> Java Build Path -> Libraries
 - ▶ dann Add Library -> JUnit -> Next klicken
 - ▶ Version auswählen
 - ▶ Finish

*mit Eclipse

Testklassen

- ▶ Sind normale Java-Klassen
- ▶ Werden nicht im gleichen Ordner wie die anderen Klassen aus dem Projekt gespeichert
 - ▶ Weil diese Testklassen nur während der Entwicklung verwendet werden
 - ▶ später beim Generieren der JAR-Datei kann man den Testordner mit den Tests weglassen
 - ▶ Optimal:
 - ▶ Testordner liegt parallel zum Source-Ordner und hat die gleichen Unterordner
 - ▶ die Testklasse hat automatisch Zugriff auf Methoden der zu testenden Klasse, die **protected** sind, weil Testklasse und zu testende Klasse im gleichen Paket liegen.

Testordner*

- ▶ separaten Ordner für die Tests mit folgenden Schritten anlegen:
 - ▶ im Package Explorer mit der rechten Maustaste auf den Projektnamen
 - ▶ Build Path -> Create New Folder... (oder New > Source Folder)
 - ▶ „tests“ als Ordner eingeben -> Finish

*mit Eclipse

Testklasse anlegen*

- ▶ im Package Explorer mit der rechten Maustaste auf den Dateinamen einer zu testenden Klasse New -> JUnit Test Case
- ▶ Stellen Sie die JUnit Version auf New JUnit 4 test
- ▶ Wählen Sie danach das Verzeichnis tests aus
- ▶ folgende Methoden auswählen, die dann mit erzeugt werden:
 - ▶ setUpBeforeClass(): Wird vor der eigentlichen Test-Klasse ausgeführt.
 - ▶ setUp(): Wird vor jeder Test-Methode in der Test-Klasse aufgerufen. (Wird am häufigsten benutzt. Kann etwa zur Initialisierung von Variablen genutzt werden.)
 - ▶ tearDownAfterClass(): Wird nach der eigentlichen Test-Klasse ausgeführt.
 - ▶ tearDown(): Wird nach jeder Test-Methode in der Test-Klasse aufgerufen.
- ▶ In Übersicht mit den möglichen Test-Methoden die zu testenden Methoden auswählen
- ▶ Mit Finish wird die Test-Klasse erstellt.

*mit Eclipse

New JUnit Test Case

JUnit Test Case

Select the name of the new JUnit test case. You have the options to specify the class under test and on the next page, to select methods to be tested.

☐ New JUnit 3 test ☒ New JUnit 4 test

Source folder:

Package:

Name:

Superclass:

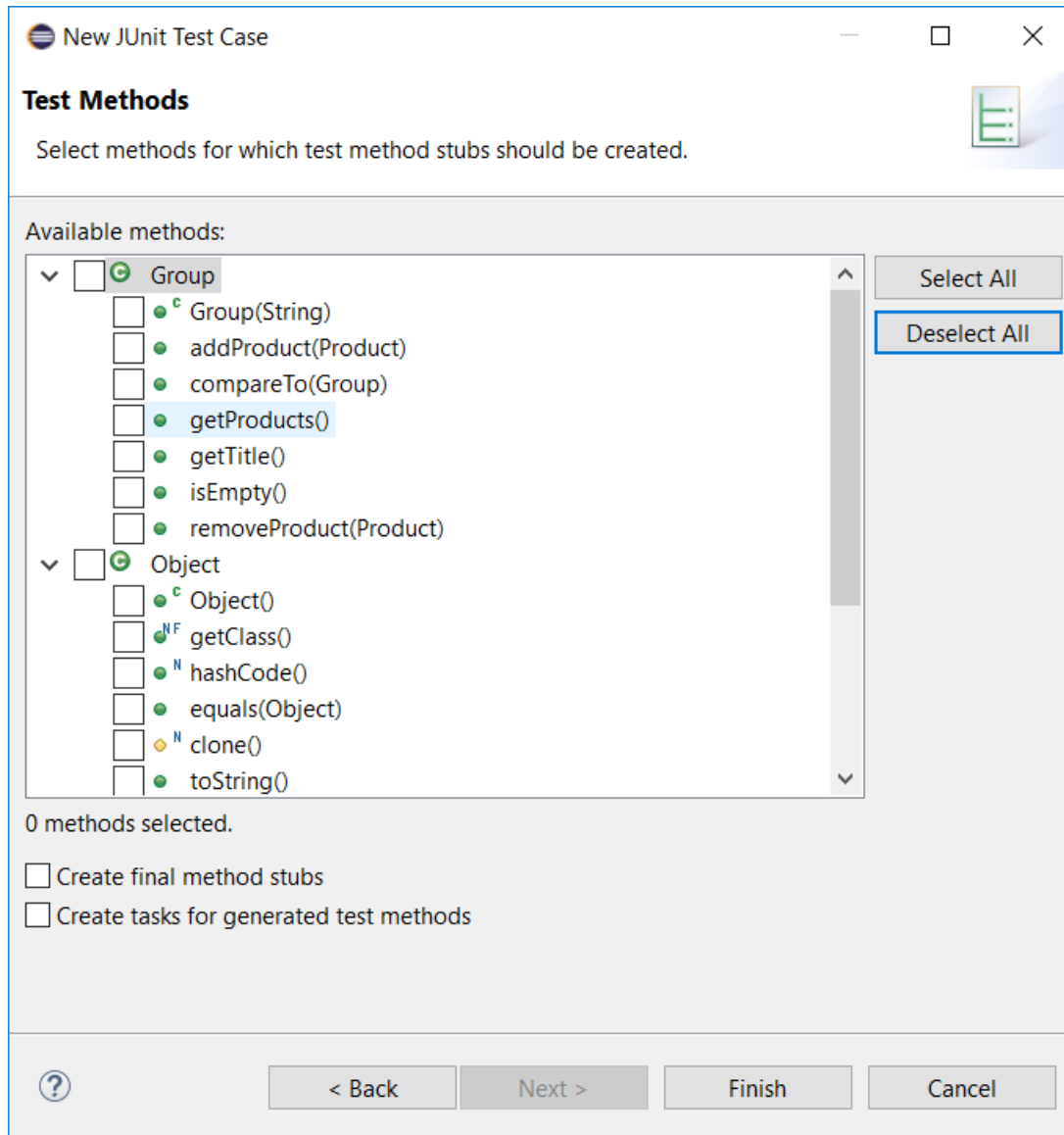
Which method stubs would you like to create?

☐ setUpBeforeClass() ☐ tearDownAfterClass()
☒ setUp() ☐ tearDown()
☐ constructor

Do you want to add comments? (Configure templates and default value [here](#))

☐ Generate comments

Class under test:




```

1 package model;
2
3 import static org.junit.Assert.*;
4
5
6
7
8 public class GroupTest {
9
10     @Before
11     public void setUp() throws Exception {
12     }
13
14     @Test
15     public void testGroup() { ges Ausschneiden
16         fail("Not yet implemented");
17     }
18
19     @Test
20     public void testAddProduct() {
21         fail("Not yet implemented");
22     }
23
24     @Test
25     public void testCompareTo() {
26         fail("Not yet implemented");
27     }
28
29     @Test
30     public void testGetProducts() {
31         fail("Not yet implemented");
32     }
33
34     @Test
35     public void testGetTitle() {
36         fail("Not yet implemented");
37     }
38
39     @Test
40     public void testIsEmpty() {
41         fail("Not yet implemented");

```

Beispiel – Aufbau Testklasse mit Testmethode

```
package model;
import static org.junit.Assert.*;
import org.junit.Before;
import org.junit.Test;
```

In org.junit.Assert gibt es Methoden, über die bestimmte Zusicherungen definiert werden können.

```
public class GroupTest {
    private Group group;
    private Product product0;
    /**
     * Erzeugt vor jeder Test-Methode eine neue Testumgebung.
     * @throws Exception
     *      Mögliche Exceptions beim setUp.
     */
    @Before
    public void setUp() throws Exception {
        //...
    }
    /**
     * Testet die verwendete Vergleichsmethode für die Sortierung der Gruppenliste.
     */
    @Test
    public void testCompareTo() {
        // ...
    }
}
```

@Before sagt, dass diese Methode vor jeder anderen Testmethode in dieser Testklasse aufgerufen werden soll.

Über die Annotation @Test werden die Testmethoden festgelegt, die während des Ausführens der Testklasse gestartet werden

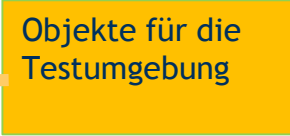
Beispiel – Testumgebung

```
public class GroupTest {
    private static final String DESCRIPTION = "description";
    private static final String TITLE = "Test-Group";
    private Group group;
    private Product product0;

    /**
     * Erzeugt vor jeder Test-Methode eine neue Testumgebung.
     * @throws Exception
     *      Mögliche Exceptions beim setUp.
     */

    @Before
    public void setUp() throws Exception {
        group = new Group(TITLE);
        product0 = new Product("Produkt0", DateFactory.now(), DESCRIPTION);
    }

    @Test
    public void testCompareTo() {
        // ...
    }
}
```



Beispiel – Testmethode

Rückgabewert von compareTo()	Bedeutung
0	Die Objekte sind bezüglich ihrer Eigenschaft(en) „gleich groß“.
-1	Das aktuelle Objekt ist „kleiner“ als das andere Objekt.
1	Das aktuelle Objekt ist „größer“ als das andere Objekt.

```
public class GroupTest {
    @Test
    public void testCompareTo() {
        // Vorbereitung
        int numberOfElements = 5;
        ArrayList<Group> groups = new ArrayList<Group>();
        groups.add(new Group("Gruppe0"));
        Group groupobject = new Group("Gruppe0");
        for (int index = 1; index < numberOfElements; index++) {
            groups.add(new Group("Gruppe" + index));
        }

        // Test:
        for (int i = 1; i < groups.size(); i++) {
            // A1 < A2
            assertEquals(-1, groups.get(i - 1).compareTo(groups.get(i)));
            // A1 > A2
            assertEquals(1, groups.get(i).compareTo(groups.get(i - 1)));
        }
        // Test: A1 = A2
        assertEquals(0, groups.get(0).compareTo(groupobject));
        for (int i = 0; i < groups.size(); i++) {
            assertEquals(0, groups.get(i).compareTo(groups.get(i)));
        }
    }
}
```

Eine alphabetisch sortierte Liste

Prüft, ob zwei Objekte gleich sind.

Erwarteter Wert -1

Tatsächlicher Wert

Zusicherungen

Methode	Beschreibung
<code>assertArrayEquals()</code>	Prüft, ob zwei Arrays die gleichen Werte enthalten.
<code>assertEquals()</code>	Prüft, ob zwei Objekte gleich sind.
<code>assertFalse()</code>	Prüft, ob eine Boolesche Bedingung <code>false</code> ist.
<code>assertNotNull()</code>	Prüft, ob ein Objekt ungleich <code>null</code> ist.
<code>assertNull()</code>	Prüft, ob ein Objekt gleich <code>null</code> ist.
<code>assertSame()</code>	Prüft, ob zwei Objekte identisch sind.
<code>assertTrue()</code>	Prüft, ob eine Boolesche Bedingung <code>true</code> ist.

prüfen, ob der übergebene Boolesche Ausdruck `true` oder `false` ist.

Prüfen ob eine Objektreferenz `null` bzw. nicht `null` ist.

Package Explorer Type Hierarchy JUnit

Finished after 0,115 seconds

Runs: 1/1 Errors: 0 Failures: 0

> model.GroupTest [Runner: JUnit 4] (0,020 s)

Failure Trace

```

10 import control.DateFactory;
11
12 public class GroupTest {
13     private static final String DESCRIPTION = "description";
14     private static final String TITLE = "Test-Group";
15     private Group group;
16     private Product product0;
17
18     /**
19      * Erzeugt vor jeder Test-Methode eine neue Testumgebung.
20      *
21      * @throws Exception
22      *     Mögliche Exceptions beim setUp.
23      */
24
25     @Before
26     public void setUp() throws Exception {
27         group = new Group(TITLE);
28         product0 = new Product("Produkt0", DateFactory.now(), DESCRIPTION);
29     }
30
31     // @Test
32     // public void testGroup() {
33     //     fail("Not yet implemented");
34     // }
35     //
36     // @Test
37     // public void testAddProduct() {
38     //     fail("Not yet implemented");
39     // }
40
41     /**
42      * Testet die verwendete Vergleichsmethode für die Sortierung der Gruppen-Arrayliste.
43      */
44     @Test
45     public void testCompareTo() {
46         // Vorbereitung
47         int numberOfElements = 5;
48         ArrayList<Group> groups = new ArrayList<Group>();
49         groups.add(new Group("Gruppe0"));
50         Group groupobject = new Group("Gruppe0");
51         for (int index = 1; index < numberOfElements; index++) {
52             groups.add(new Group("Gruppe" + index));
53         }
54         // Test:
55         for (int i = 1; i < groups.size(); i++) {
56             // A1 < A2
57             assertEquals(-1, groups.get(i - 1).compareTo(groups.get(i)));
58             // A1 > A2
59             assertEquals(1, groups.get(i).compareTo(groups.get(i - 1)));
60         }
61         // Test: A1 = A2
62         assertEquals(0, groups.get(0).compareTo(groupobject));
63         for (int i = 0; i < groups.size(); i++) {
64             assertEquals(0, groups.get(i).compareTo(groups.get(i)));
65         }
66     }

```

Package ExplorerType HierarchyJUnit

Finished after 0,114 seconds

Runs: 1/1Errors: 0Failures: 1

model.GroupTest [Runner: JUnit 4] (0,031 s)

testCompareTo (0,031 s)

Failure Trace

```
java.lang.AssertionError: expected:<1> but was:<-1>
    at org.junit.Assert.fail(Assert.java:88)
    at org.junit.Assert.failNotEquals(Assert.java:834)
    at org.junit.Assert.assertEquals(Assert.java:645)
    at org.junit.Assert.assertEquals(Assert.java:631)
    at model.GroupTest.testCompareTo(GroupTest.java:57)
    at sun.reflect.NativeMethodAccessorImpl.invoke(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(Unknown Source)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(Unknown Source)
    at java.lang.reflect.Method.invoke(Unknown Source)
    at org.junit.runners.model.FrameworkMethod$1.runReflectiveCall(FrameworkMethod$1.java:50)
    at org.junit.runners.model.ReflectiveCallable.run(ReflectiveCallable.java:12)
    at org.junit.runners.model.FrameworkMethod.invokeExplosively(FrameworkMethod.java:143)
    at org.junit.internal.runners.statements.InvokeMethod.evaluate(InvokeMethod.java:17)
    at org.junit.internal.runners.statements.RunBefores.evaluate(RunBefores.java:26)
    at org.junit.runners.ParentRunner.runLeaf(ParentRunner.java:325)
    at org.junit.runners.BlockJUnit4ClassRunner.runChild(BlockJUnit4ClassRunner.java:78)
    at org.junit.runners.BlockJUnit4ClassRunner.runChild(BlockJUnit4ClassRunner.java:57)
    at org.junit.runners.ParentRunner$3.run(ParentRunner.java:290)
    at org.junit.runners.ParentRunner$1.schedule(ParentRunner.java:71)
    at org.junit.runners.ParentRunner.runChildren(ParentRunner.java:288)
    at org.junit.runners.ParentRunner.access$0(ParentRunner.java:58)
    at org.junit.runners.ParentRunner$1.evaluate(ParentRunner.java:269)
```

Group.javaProduct.javaMain.javaModel.javaViewGUL.javaControllerG...DateFactory...

```
10 import control.uateractory;
11
12 public class GroupTest {
13     private static final String DESCRIPTION = "description";
14     private static final String TITLE = "Test-Group";
15     private Group group;
16     private Product product0;
17
18     /**
19      * Erzeugt vor jeder Test-Methode eine neue Testumgebung.
20      *
21      * @throws Exception
22      *     Mögliche Exceptions beim setUp.
23      */
24
25     @Before
26     public void setUp() throws Exception {
27         group = new Group(TITLE);
28         product0 = new Product("Produkt0", DateFactory.now(), DESCRIPTION);
29     }
30
31     // @Test
32     // public void testGroup() {
33     //     fail("Not yet implemented");
34     // }
35
36     // @Test
37     // public void testAddProduct() {
38     //     fail("Not yet implemented");
39     // }
40
41     /**
42      * Testet die verwendete Vergleichsmethode für die Sortierung der Gruppen-Arrayliste.
43      */
44     @Test
45     public void testCompareTo() {
46         // Vorbereitung
47         int numberOfElements = 5;
48         ArrayList<Group> groups = new ArrayList<Group>();
49         groups.add(new Group("Gruppe0"));
50         Group groupobject = new Group("Gruppe0");
51         for (int index = 1; index < numberOfElements; index++) {
52             groups.add(new Group("Gruppe" + index));
53         }
54         // Test:
55         for (int i = 1; i < groups.size(); i++) {
56             // A1 < A2
57             assertEquals(1, groups.get(i - 1).compareTo(groups.get(i)));
58             // A1 > A2
59             assertEquals(1, groups.get(i).compareTo(groups.get(i - 1)));
60         }
61         // Test: A1 = A2
62         assertEquals(0, groups.get(0).compareTo(groupobject));
63         for (int i = 0; i < groups.size(); i++) {
64             assertEquals(0, groups.get(i).compareTo(groups.get(i)));
65         }
66     }
67 }
```

<terminated> GroupTest [JUnit] C:\Program Files\Java\jre1.8.0_111\bin\javaw.exe (26.04.2017, 12:44:22)

ProblemsJavadocDeclarationSearchConsoleModel ExplorerPropertiesModel Validation

Dr. Inform. Corina Kopka

Programmieren in Java

15

Literatur

- ▶ JUnit Testframework Homepage. <http://www.junit.org/>
- ▶ <http://www.vogella.com/tutorials/JUnit/article.html>
- ▶ <http://www.frankwestphal.de/UnitTestingmitJUnit.html>
- ▶ Beispielprojekt in Confluence
- ▶ Weiteres Beispielprojekt in Confluence unter Shared Links:
 - ▶ <https://sopra.cs.tu-dortmund.de/wiki/projekte/beispiele/gettingthingsdone/aufwaermprojekt/start>