Mocking mit Mockito

Carsten Gips (HSBI)

Motivation: Entwicklung einer Studi-/Prüfungsverwaltung

- Team A:

```
public class Studi {
    String name;  LSF lsf;

    public Studi(String name, LSF lsf) {
        this.name = name;  this.lsf = lsf;
    }

    public boolean anmelden(String modul) { return lsf.anmelden(name, modul); }
    public boolean einsicht(String modul) { return lsf.ergebnis(name, modul) > 50; }
}
```

- Team B:

```
public class LSF {
    public boolean anmelden(String name, String modul) { throw new UnsupportedOperationException(); }
    public int ergebnis(String name, String modul) { throw new UnsupportedOperationException(); }
}
```

Wie kann Team A seinen Code testen?

Manuell Stubs implementieren

```java
public class StudiStubTest {
    Studi studi;  LSF lsf;

    @Before
    public void setUp() { lsf = new LsfStub();  studi = new Studi("Harald", lsf); }

    @Test
    public void testAnmelden() { assertTrue(studi.anmelden("PM-Dungeon")); }
    @Test
    public void testEinsicht() { assertTrue(studi.einsicht("PM-Dungeon")); }


    // Stub für das noch nicht fertige LSF
    class LsfStub extends LSF {
        public boolean anmelden(String name, String modul) { return true; }
        public int ergebnis(String name, String modul) { return 80; }
    }
}
```

Demo hsbi.StudiStubTest

Mockito: Mocking von ganzen Klassen

```java
public class StudiMockTest {
    Studi studi;  LSF lsf;

    @Before
    public void setUp() { lsf = mock(LSF.class);  studi = new Studi("Harald", lsf); }

    @Test
    public void testAnmelden() {
        when(lsf.anmelden(anyString(), anyString())).thenReturn(true);
        assertTrue(studi.anmelden("PM-Dungeon"));
    }

    @Test
    public void testEinsichtI() {
        when(lsf.ergebnis("Harald", "PM-Dungeon")).thenReturn(80);
        assertTrue(studi.einsicht("PM-Dungeon"));
    }
    @Test
    public void testEinsichtII() {
        when(lsf.ergebnis("Harald", "PM-Dungeon")).thenReturn(40);
        assertFalse(studi.einsicht("PM-Dungeon"));
    }
}
```

Mockito: Spy = Wrapper um ein Objekt

```java
public class StudiSpyTest {
    Studi studi;  LSF lsf;

    @Before
    public void setUp() { lsf = spy(LSF.class);  studi = new Studi("Harald", lsf); }

    @Test
    public void testAnmelden() { assertTrue(studi.anmelden("PM-Dungeon")); }

    @Test
    public void testEinsichtI() {
        doReturn(80).when(lsf).ergebnis("Harald", "PM-Dungeon");
        assertTrue(studi.einsicht("PM-Dungeon"));
    }
    @Test
    public void testEinsichtII() {
        doReturn(40).when(lsf).ergebnis("Harald", "PM-Dungeon");
        assertFalse(studi.einsicht("PM-Dungeon"));
    }
}
```

Demo hsbi.StudiSpyTest

Wurde eine Methode aufgerufen?

```java
public class VerifyTest {
    @Test
    public void testAnmelden() {
        LSF lsf = mock(LSF.class);  Studi studi = new Studi("Harald", lsf);

        when(lsf.anmelden("Harald", "PM-Dungeon")).thenReturn(true);

        assertTrue(studi.anmelden("PM-Dungeon"));


        verify(lsf).anmelden("Harald", "PM-Dungeon");
        verify(lsf, times(1)).anmelden("Harald", "PM-Dungeon");

        verify(lsf, atLeast(1)).anmelden("Harald", "PM-Dungeon");
        verify(lsf, atMost(1)).anmelden("Harald", "PM-Dungeon");

        verify(lsf, never()).ergebnis("Harald", "PM-Dungeon");

        verifyNoMoreInteractions(lsf);
    }
}
```

Demo hsbi.VerifyTest

Fangen von Argumenten

```java
public class MatcherTest {
    @Test
    public void testAnmelden() {
        LSF lsf = mock(LSF.class);  Studi studi = new Studi("Harald", lsf);

        when(lsf.anmelden(anyString(), anyString())).thenReturn(false);
        when(lsf.anmelden("Harald", "PM-Dungeon")).thenReturn(true);

        assertTrue(studi.anmelden("PM-Dungeon"));
        assertFalse(studi.anmelden("Wuppie?"));

        verify(lsf, times(1)).anmelden("Harald", "PM-Dungeon");
        verify(lsf, times(1)).anmelden("Harald", "Wuppie?");

        verify(lsf, times(2)).anmelden(anyString(), anyString());
        verify(lsf, times(1)).anmelden(eq("Harald"), eq("Wuppie?"));
        verify(lsf, times(2)).anmelden(argThat(new MyHaraldMatcher()), anyString());
    }


    class MyHaraldMatcher implements ArgumentMatcher<String> {
        public boolean matches(String s) { return s.equals("Harald"); }
    }
}
```

Ausblick: PowerMock

Mockito sehr mächtig, aber unterstützt (u.a.) keine

- Konstruktoren
- private Methoden
- final Methoden
- static Methoden

=> Lösung: PowerMock

Wrap-Up

- Gründliches Testen ist ebenso viel Aufwand wie Coden!

- Mockito ergänzt JUnit:
  - Mocken ganzer Klassen (`mock()`, `when().thenReturn()`)
  - Wrappen von Objekten (`spy()`, `doReturn().when()`)
  - Auswerten, wie häufig Methoden aufgerufen wurden (`verify()`)
  - Auswerten, mit welchen Argumenten Methoden aufgerufen wurden (`anyString`)

LICENSE