# Mockito-Kurzanleitung

Diese Anleitung bietet die wichtigsten Grundlagen zum Testen mit Mockito. Weitere Hilfe ist unter <http://site.mockito.org/mockito/docs/current/org/mockito/Mockito.html> zu finden.

|  |  |
| --- | --- |
| Mockito in Testfall importieren | **import** **static** org.mockito.Mockito.\*; |
| Ein Mock-Objekt erzeugen.  Beispiel: Ein Mock-Objekt für eine Liste von Strings. | List<String> liste = *mock*(List.**class**); |
| Antwort für den Aufruf einer Methode beim Mock-Objekt konfigurieren. | *when*(liste.get(0)).thenReturn("Meier"); |
| Beim Aufruf einer Methode des Mock-Objekts eine Exception werfen. | *when*(liste.get(1)).thenThrow(**new** RuntimeException()); |
| Überprüfen, ob Methode beim Mock-Objekt mit den erwarteten Argumenten aufgerufen worden ist.  Beispiel: Wurde die Methode get mit Argument 0 aufgerufen? | *verify*(liste).get(0); |
| Mit einem Argument-Matcher überprüfen, ob Methode beim Mock-Objekt mit irgendwelchen Argumenten aufgerufen worden ist. | *verify*(liste).get(*anyInt*()); |
| Überprüfen, ob Methode beim Mock-Objekt eine bestimmte Anzahl Male aufgerufen worden ist. | *verify*(liste, *times*(3)).get(0); |
| Reihenfolge der Methodenaufrufe beim Mock-Objekt überprüfen. | InOrder order = *inOrder*(liste);  order.verify(liste).get(0);  order.verify(liste).get(1); |

Nachfolgend sind obige Anweisungen in einem Beispiel-Test zusammengefasst. In diesem Test wird für die List-Schnittstelle aus der Java-Bibliothek eine Mock-Klasse erstellt. In einem realen Test würde diese Mock-Klasse dann für den Test einer weiteren Klasse eingesetzt. Das ist hier nicht so, da es nur darum geht die verschiedenen Möglichkeiten von Mockito aufzuzeigen. Es gibt also keine weitere zu testende Klasse in diesem Beispiel.

**import** **static** org.mockito.Mockito.\*;

**import** **static** org.junit.Assert.\*;

**import** org.junit.Test;

**import** org.mockito.InOrder;

**import** java.util.List;

**public** **class** MockitoKurzanleitung {

**private** List<String> liste = *mock*(List.**class**);

@Test(expected = RuntimeException.**class**)

**public** **void** test() {

InOrder order = *inOrder*(liste);

*when*(liste.get(0)).thenReturn("Meier");

*when*(liste.get(999)).thenThrow(**new** RuntimeException());

*assertEquals*("Meier", liste.get(0));

*assertEquals*(**null**, liste.get(1));

*verify*(liste).get(0);

*verify*(liste, *times*(1)).get(0);

*verify*(liste, *times*(2)).get(*anyInt*());

order.verify(liste).get(0);

order.verify(liste).get(1);

liste.get(999);

}