# Einführung in Stub- und Mock-Klassen

## Einleitung

In Modul 226-1 haben Sie gelernt, dass man Unit-Tests einsetzt, um eine einzelne Klasse zu testen. Dazu gibt es Werkzeuge, wie JUnit, welche es erlauben, einen Unit-Test automatisiert und beliebig oft durchzuführen. Nach jeder Änderung an der Klasse kann der Test wiederholt werden. Damit ist sichergestellt, dass die Klasse alle bisherigen Anforderungen nach wie vor erfüllt. Man spricht von *Regression-Tests*.

Oft ist es nicht möglich ein Klasse vollständig isoliert zu testen, weil sie von anderen Klassen abhängig ist. Dies ist der Fall, wenn eine Klasse Objekte anderer Klassen referenziert (über Instanzvariablen) oder wenn sie beim Aufruf ihrer Methoden Referenzen auf Objekte anderer Klassen empfängt (als Argument). In solchen Fällen müssen im Test zusätzliche Objekte bereitgestellt werden, welche nicht im Zentrum des Tests stehen. Man spricht hier von Kollaborationsobjekten (*collaborator objects*). Dies in Abgrenzung zum eigentlich zu testenden Objekt, welches oft als *system under test* (SUT) bezeichnet wird.

Im einfachsten Fall spielen Kollaborationsobjekte im Test die Rolle von reinen Dummies. Das heisst, sie werden herumgereicht, aber nicht weiter bearbeitet. Oft werden Sie aber auch in den Test mit einbezogen, indem auch bei ihnen Bedingungen überprüft werden, welche durch das SUT eingehalten werden müssen.

Schwieriger wird es, wenn das SUT darauf angewiesen ist, dass sich die Kollaborationsobjekte nach einem bestimmten Muster verhalten. Das heisst, dass die Kollaborationsobjekte beim Aufruf ihrer Methoden bestimmte, vordefinierte Resultate liefern. Das kann einiges Kopfzerbrechen geben, wenn sich hinter einem Kollaborationsobjekt ein ganzes Subsystem, wie ein Datenbank, oder ein GUI versteckt.

In diesen Situationen arbeitet man dann mit sogenannten Stubs („Stummeln“) oder gar mit Mocks (von Mock-up, Attrappe). Dies sind Klassen, welche die gleiche Schnittstelle wie das Original anbieten, dahinter aber eine auf den Test zugeschnittene Funktion implementieren.

## Stub-Klassen

Eine Stub-Klasse ist eine Klasse, welche die gleichen Methoden, wie die Originalklasse anbietet. Allerdings werden die Methoden nicht vollständig implementiert, sondern sie liefern beim Aufruf nur für den Test vorprogrammierte Resultate. Auf alles, was ausserhalb dieser Vorprogrammierung liegt kann eine Stub-Klasse nicht reagieren. Stub-Klassen können Informationen über die aufgerufenen Methoden sammeln. Diese Informationen können später im Test überprüft werden. Eine Stub-Klasse wird in der Regel von Hand für einen bestimmten Testfall programmiert und ist entsprechend eingeschränkt in ihren Möglichkeiten. Sie kann nicht ohne weiteres für andere Testfälle eingesetzt werden.

## Mock-Klassen

Eine Mock-Klasse ist eine Klasse, welche ebenfalls die gleichen Methoden, wie die Originalklasse anbietet. Sie ist aber viel flexibler als eine Stub-Klasse. Die Methoden können vor dem Test mit einem beliebigen Verhalten konfiguriert werden. Dazu werden je nach Werkzeug Regeln eingesetzt (wenn … dann …). Auch bei der Überprüfung bieten Mock-Klassen viel mehr Möglichkeiten. So kann direkt überprüft werden, ob eine bestimmte Methode wirklich mit den erwarteten Argumenten aufgerufen worden ist. Auch die Reihenfolge und Anzahl der Aufrufe kann überprüft werden. Für die Erstellung von Mock-Klassen setzt man normalerweise ein Werkzeug wie Mockito oder jMockit ein.

Zur Vertiefung wird folgender Artikel empfohlen: <http://martinfowler.com/articles/mocksArentStubs.html>