Legacy Code

Lukas Panni
TINF18B5

DHBW Karlsruhe
Vorlesung Advanced Software Engineering Semester 5/6

Inhaltsverzeichnis

1 Abhängigkeiten brechen

Im folgenden sollen einige Beispiele aufgeführt werden, bei denen Abhängigkeiten mithilfe der in der Vorlesung behandelten Techniken beseitigt werden. Ohne diese Abhängigkeiten sind Tests leichter zu entwickeln, sodass weitere Änderungen, zum Beispiel zur Verbesserung des Designs, später einfacher und komfortabler möglich sind. Das Ziel beim Brechen der Abhängigkeiten ist es, die Testbarkeit der betroffenen Klassen beziehungsweise Methoden zu erhöhen.

1.1 ExtractInterface bei Commit 1e47e7b

Ausgangszustand

Die Klasse *AuthHandler* ist stark abhängig von der Android-Klasse *Activity* und kann auch nicht erstellt werden ohne eine Instanz dieser Klasse der *getInstance*-Methode zu übergeben. Die Testbarkeit ist schlecht, da eine Android-*Activity*-Instanz nicht ohne weiteres im Test-Kontext erstellt werden kann. Auch ein Fake-Objekt ist schwer zu erstellen, da eine finale Methode genutzt wird, in Ableitungen von *Activity* nicht überschrieben werden kann. Da die Klasse *AuthHandler* für die OAuth-Authentifizierung verantwortlich ist und damit wichtig für das Gesamtsystem, ist es wichtig, Tests für diese Klasse zu ermöglichen.

Gewählte Technik

Die *getInstance*-Methode benötigt eine Instanz der Klasse *Activity*, die im Test-Kontext nicht leicht erstellbar ist und für die auch Fake-Objekte nur schwer erstellt werden können. Da allerdings nur wenige Methoden der *Activity*-Klasse verwendet werden, und eine Änderung der *AuthHandler* Klasse nur vergleichsweise wenige Änderungen erfordert, kann in diesem Fall **ExtractInterface** angewendet werden. So wird die Abhängigkeit von *AuthHandler* zu *Activity* gelöst, indem das Interface *AuthHandlerActivity* eingeführt und der Parameter von *getInstance* angepasst wird, sodass eine Instanz vom Typ des Interfaces verwendet wird. Dabei werden auch kleinere Anpassungen an Methodenaufrufen vorgenommen, sodass Methoden des Interfaces aufgerufen werden. **ExtractInterface** ist durch IDE-Unterstützung vergleichsweise einfach und ohne große Fehleranfälligkeit durchzuführen. Außerdem wird die Abstraktion durch diese Technik verbessert.

Endzustand

AuthHandler nutzt nach dieser Änderung nur noch die Methoden des Interfaces, wodurch die Testbarkeit erhöht wird. Allerdings gibt es hier das Problem, dass *AuthHandler* die Third-Party-Klasse *AuthorizationService* verwendet, die auf eine *Activity*-Instanz angewiesen ist. Deshalb gibt es im neu eingeführten Interface eine Methode, die eine *Activity* zurückgibt. Die Abhängigkeit konnte also in diesem Fall nicht komplett aufgelöst werden. Trotzdem wurde die Testbarkeit erhöht und die Abstraktion verbessert. Diese Änderung stellt damit eine gute Basis für weitere Refactorings und Änderungen dar, wie zum Beispiel in Commit cbc5318.

1.2 Parametrize Constructor bei Commit d098b93

Ausgangszustand

Die Klassen *RepositoryDataRepository* und *UserContributionsRepository* haben eine Abhängigkeit zur Klasse *ResponseCache* und erzeugen im Konstruktor eine Instanz dieser Klasse und speichern diese in einer Instanzvariablen. Für den Test der Klassen ist es hilfreich, diese Instanzvariable durch eine eigene Cache-Implementierung zu nutzen. In diesem Fall lässt sich die Abhängigkeit mit Hilfe von **Parametrize Constructor** brechen.
Dazu wird ein neuer Konstruktor erzeugt, der einen zusätzlichen Parameter vom Typ der zu überschreibenden Instanzvariable entgegennimmt. In diesem Beispiel eine Instanz von *ResponseCache*. Die existierenden Konstruktoren rufen den neuen Konstruktor mit dem ursprünglichen Wert der Variable auf. Dabei bleibt die alte Funktionalität erhalten und es müssen keine Anpassungen an anderen Stellen erfolgen.

Endzustand

Durch das Brechen der Abhängigkeit ist es nun möglich, beim Erstellen einer Instanz vom Typ *RepositoryDataRepository* oder *UserContributionsRepository* ein *ResponseCache*-Objekt zu übergeben. Dadurch kann zum Beispiel beim Test ein Fake-Objekt übergeben werden.

1.3 ExtractInterface bei Commit da964f5

Ausgangszustand

Ein Objekt der Klasse *AuthHandler* wird zum Beispiel in der Klasse *GHClient* (bei da964f5 umbenannt in *GithubOAuthClient*) benötigt, um Daten über die API abrufen zu können. Aktuell ist ein Test dieser Komponente nur schwer möglich, da *AuthHandler* selbst wiederum Abhängigkeiten zu einer Third-Party Bibliothek besitzt. Deshalb ist hier das erstellen von Fake-/Mock-Objekten sehr aufwändig. Bei 1.1 wurde bereits die Testbarkeit der *AuthHandler*-Klasse erhöht und durch ExtractInterface an dieser Stelle soll die Testbarkeit der von *AuthHandler* abhängigen Klassen erhöht werden.

Gewählte Technik

Es wurde **ExtractInterface** gewählt um Abhängigkeiten zu *AuthHandler* zu beseitigen. Dazu wurde das Interface *AuthenticationHandler* eingeführt wird, das in der alten *AuthHandler*-Klasse (zur besseren Verständlichkeit umbenannt zu *GithubOAuthHandler*) implementiert wird. Um den Vorgang zu ermöglichen wurde zuvor die Singleton-Eigenschaft entfernt (13efee8) und die Methoden und ihre Verwender wurden angepasst (473384c). Anschließend kann das Interface erstellt werden und bei allen Verwendern der Ursprünglichen Klasse, die dies erlauben, die Abhängigkeit zur Konkreten Klasse durch eine Abhängigkeit zum Interface ersetzt werden.

Endzustand

In der Klasse *GithubOAuthClient* konnte die Abhängigkeit zu *GithubOAuthHandler* (vormals *AuthHandler* vollständig beseitigt werden. Dadurch werden Tests der Klasse mit Fake-/Mock-Implementierungen des neuen Interfaces *AuthenticationHandler* möglich. Außerdem wird die Abstraktion verbessert und die konkrete Authentifizierungs-Klasse könnte leichter ausgetauscht werden, was zum Beispiel benötigt wird, wenn ein anderes OAuth-Framework eingesetzt werden sollte.

1.4 Parametrize Method bei Commit d8c995c

Ausgangszustand

Die Klasse *TimeSpanFactory*, die verwendet wird um *TimeSpan*-Objekte zu erzeugen hat eine starke Abhängigkeit zur Java-Klasse *java.util.Calendar*. In den Methoden von TimeSpanFactory wird jeweils eine Calendar-Instanz benötigt. Um diese zu erhalten wird die Methode *Calendar.getInstance()* verwendet. Da die verwendete Calendar-Instanz das Ergebnis der Methoden beeinflusst muss die Instanz für Tests ausgetauscht werden können. Dies ist aber aktuell nicht möglich.

Gewählte Technik

Um die Abhängigkeit der Methoden zu *Java.util.Calendar* zu beseitigen wurde die Technik **Parametrize Method** gewählt. Bei dieser Technik wird für die Methode, bei der die Abhängigkeit gebrochen werden muss, eine neue Methode mit einem zusätzlichen Parameter angelegt. Dieser Parameter ersetzt dann die Variable, die im Test benötigt wird. Die eigentliche Funktionalität der alten Methode wird in die neue Methode verschoben und die alte Methode ruft diese mit dem ursprünglichen Wert der Variablen auf. So sind keine Änderungen an bestehendem Code notwendig und trotzdem wird ein Test der Methode ermöglicht.

Endzustand

Die Technik Parametrize Method wurde für mehrere Methoden in TimeSpanFactory angewendet um die Abhängigkeit zu lösen, sodass Tests nun möglich sind. Durch die Möglichkeit eine Calendar-Instanz zu übergeben können die Methoden einfach getestet werden.

1.5 Subclass and Override Method bei Commit 13676e5

Ausgangszustand

Die Klasse *GithubOAuthClient* verwendet eine Third-Party-Library um GraphQL-Anfragen an die GitHub-API durchzuführen. Diese Abhängigkeit macht Tests vergleichsweise schwierig. Um also Tests durchführen zu können ist es notwendig, diese Abhängigkeit zu lösen.

Gewählte Technik

Um diese Abhängigkeit zu lösen wurde die Technik **Subclass and Override Method** gewählt. Diese Technik erlaubt es, eine Methode, die im Test nicht sichtbar ist, zu ändern. So kann zum Beispiel der Rückgabewert der Methode für den Test festgelegt werden. Dafür wird zunächst die bestehende Methode umbenannt und eine neue Methode mit der gleichen Signatur der eben umbenannten Methode angelegt. Diese neue Methode muss für Ableitungen der Klasse sichtbar sein und kann dort überschrieben werden. In der Basisklasse werden alle Aufrufe der alten Methode durch Aufrufe der neuen Methode ersetzt, die selbst wiederum die alte Methode aufruft.
Konkret wurde die private Methode *GithubOAuthClient.getGraphqlClient* umbenannt zu *GithubOAuthClient.getGraphqlClientInternal* und eine neue protected Methode *GithubOAuthClient.getGraphqlClient* erstellt. So ist es möglich in Ableitungen der Klasse *GithubOAuthClient* das Verhalten der Methode *GithubOAuthClient.getGraphqlClient* zu verändern.

Endzustand

Durch Subclass and Override Method wurde es möglich gemacht, die Klasse *GithubOAuthClientTestImplementation* zu erstellen, die *getGraphqlClient* so überschreibt, dass eine Mock-Instanz zurückgegeben werden kann. So wird die Abhängigkeit zur Third-Party-Library reduziert und die Entwicklung von Tests fällt leichter.