Softwaretechnische Grundlagen
Programmierpraktikum

Florian Dittrich
Sommersemester 2016

Projekt #7

TDDT (Test Driven Development Trainer)

`package logik`

Inhaltsverzeichnis

class Logik

Zuständigkeit

Die Klasse Logik beinhaltet den Testzyklus, der die einzelnen Phasen RED, GREEN und REFACTOR enthält, wobei REFACTOR in zwei Schritte unterteilt werden kann. Sie ist zuständig für den Zyklusablauf und unterrichtet die grafische Benutzeroberfläche über den aktuellen Schritt, damit sie entsprechende Eingaben zulassen oder verweigern kann. Die Klasse Logik enthält selbst keine Ausführungsschleife, sondern wird von der grafischen Oberfläche aufgerufen, den Test durchführen zu lassen und dann gegebenfalls einen Zyklusschritt weiter zu gehen. Dabei besteht die Möglichkeit, einen zweiten REFACTOR-Schritt zuzulassen, in dem auch die Test optimiert werden dürfen. Auch das Ausführen eines Abschlusstests, der bestanden werden muss, ist möglich. Dazu werden der Klasse Logik Instanzen der Klassen Testen und Konfig übergeben, die dann aufgerufen werden können. Aus Konfig werden die relevanten Einstellungen ausgelesen, Testen führt die Tests aus.

Die Klasse LogikRueckgabe verwaltet die Rückgabe der Klasse Logik an die grafische Benutzeroberfläche. Dazu stellt sie zunächst die Funktion wurdeBestanden zur Verfügung, nach deren Rückgabewert Textfelder gesperrt oder freigegeben werden können.

Methoden

void Logik(ITester tester, IKonfig konfig)

Beim Erstellen einer Instanz von Logik werden ein Objekt übergeben, das das Interface ITester implementiert und eines, das IKonfig erfüllt.

ITestenRueckgabe weiter()

Beim Ausführen dieser Methode wird versucht, einen Schritt im Zyklus weiter zu gehen, indem ein Test ausgeführt und das Ergebnis in einem Objekt zurückgegeben wird, das das Interface ITestenRueckgabe erfüllt. Wurden alle Tests erfüllt, wird einen Schritt im Zyklus weiter gegangen.

void abbrechen()

Das Ausführen dieser Methode setzt die gesamte Arbeit seit dem letzten bestandenen Test auf den ersten Schritt im Entwicklungszyklus zurück.

void zuruecksetzen()

Das Ausführen dieser Methode setzt die gesamte Arbeit seit dem letzten bestandenen Akzeptanztest auf den ersten Schritt im Entwicklungszyklus zurück.

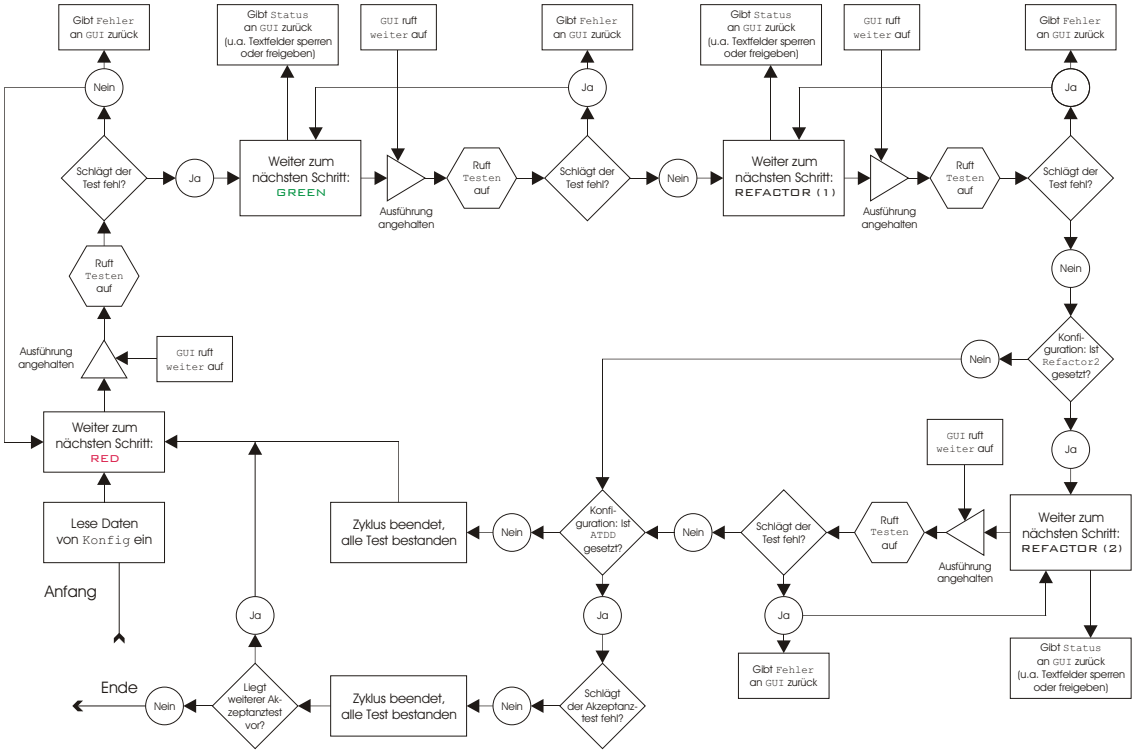
Abhängigkeiten von anderen Klassen und Interfaces

Die Klasse `Logik` ist von folgenden Klassen und Interfaces abhängig:

- interface ITester
- interface ILogikKonfig
- interface ILogikKonfigDaten
- interface ITestenRueckgabe

Flussdiagramm

Das folgende Flussdiagramm veranschaulicht die Funktionsweise eine Instanz von `Logik`. Nach der Initialisierung werden die relevanten Einstellungen aus der Konfiguartionsdatei geladen. Diese sind `ATDD` und `Refactor2`. Anschließend wird jeder Zyklus durchlaufen, indem der `GUI` die Datenstruktur ausgegeben wird, mit der sie dann Textfelder sperrt oder freigibt. Sie ruft nach erfolgter Eingabe des Nutzer die Funktion `weiter()` der Klasse `Logik` auf, die dann `Tester` aufruft und das Ergebnis in einer Datenstruktur erhält, die bei einem Fehlschlag an `GUI` zurückgegeben wird. Bei Erfolg wird zusätzlich der nächste Zyklusschritt durchgeführt. Wurde der Schritt `REFACTOR(1)` abgeschlossen und der Wert `Refactor` ist nicht gesetzt und `REFACTOR(2)` wird übersprungen. Falls `ATDD` gestetzt ist, wird nach beendigung des Zyklus noch der Akzeptanztest durchgeführt. Ist danach kein weiterer Akzeptanztest vorhanden, wird der Entwicklungszyklus beendet. Ist `ATDD` nicht gesetzt, wird stets zum ersten Zyklusschritt `RED` gesprungen.


Anfang
Lese Daten von Konfig ein
Weiter zum nächsten Schritt: RED
GUI ruft weiter auf
Ausführung angehalten
Ruft Testen auf
Schlägt der Test fehl?
Nein
Ja
Gibt Fehler an GUI zurück
Weiter zum nächsten Schritt: GREEN
Gibt Status an GUI zurück (u.a. Textfelder sperren oder freigeben)
Weiter zum nächsten Schritt: REFACTOR (1)
Konfi-guration: ist Refactor2 gesetzt?
Weiter zum nächsten Schritt: REFACTOR (2)
Konfi-guration: ist ATDD gesetzt?
Schlägt der Akzeptanz-test fehl?
Zyklus beendet, alle Test bestanden
Liegt weiterer Ak-zeptanztest vor?
Ende

interface ITestenRueckgabe

Zuständigkeit

Das Interface `ITestenRueckgabe` fordert eine Funktion `boolean testBestanden()`, an der die Klasse `Logik` ablesen kann, ob alle Tests bestanden wurden oder nicht. Wurden alle Tests bestanden, kann `Logik` mit dem Testzyklus fortfahren.

Methoden

`public boolean testBestanden()`

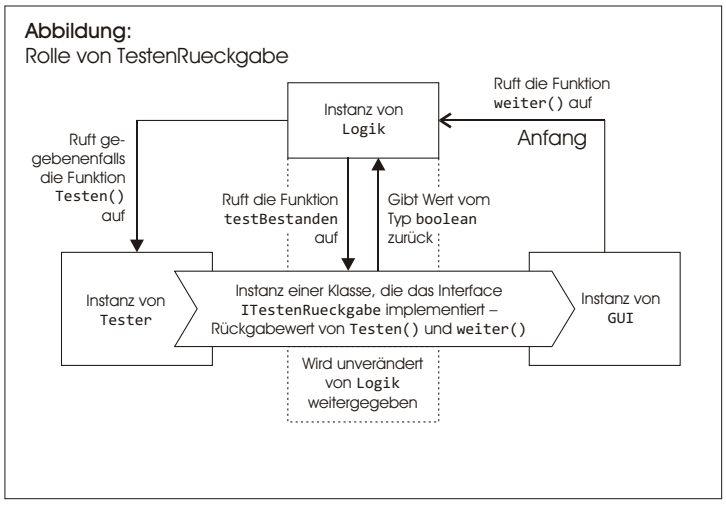
Diese Methode gibt als Wahrheitswert zurück, ob alle Tests in der Durchführung bestanden wurden, aus das Objekt stammt, dass `ITestenRueckgabe` implementiert.

Abhängigkeiten von anderen Klassen und Interfaces

Dieses Interface besitzt keine Abhängigkeiten von anderen Klassen oder Interfaces.

Flussdiagramm

Alle Instanzen einer Klasse die `ITestenRueckgabe` implementiert, dienen als Wrapper und werden von `Testen` zurückgegeben. Sie werden von Logik unverändert an die `GUI` weitergegeben, wobei nur die Funktion `testBestanden()` aufgerufen wird, die einen Wahrheitswert zurückgibt.



`interface ITestenUebergabe`

Zuständigkeit

Klassen, die dieses Interface erfüllen, werden von `GUI` an `Logik` weitergegeben, um damit die Funktion `Testen` einer Klasse aufzurufen, die `ITester` implementiert.

Methoden

Bisher keine

Abhängigkeiten von anderen Klassen und Interfaces

Dieses Interface besitzt keine Abhängigkeiten von anderen Klassen oder Interfaces.

interface ITester

Zuständigkeit

Eine Klasse, die dieses Interface implementiert, soll für `ILogik` einen Test ausführen und das Ergebnis in einem Objekt zurückgegeben, dass ITestenRueckgabe implementiert.

Methoden

`public ITestenRueckgabe Testen(ITestenUebergabe quelltext)`

Diese Methode testen den in `ITestenUebergabe` übergebenen Quelltext und gibt das Ergebnis in `ITestenUebergabe` zurück. Tritt ein Fehler auf, wird `null` gegeben.

Abhängigkeiten von anderen Klassen und Interfaces

Die Klasse `Logik` Klassen und Interfaces abhängig:

- interface ITestenRueckgabe
- interface ITestenUebergabe

interface ILogikKonfigDaten

Zuständigkeit

Alle Klassen, die dieses Interface implementieren, sollen für `Logik` eine Funktion `boolean toBoolean()` zur Verfügung stellen, die einen Wahrheitswert zur zugeordneten Eigenschaft zurückgibt und so ihre Verarbeitung ermöglicht.

Methoden

`public boolean toBoolean()`

Diese Methode gibt einen Wahrheitswert zurück, der der entsprechenden Einstellung entspricht. Im Zweifelsfall wird `false` zurückgegeben.

Abhängigkeiten von anderen Klassen und Interfaces

Dieses Interface besitzt keine Abhängigkeiten von anderen Klassen oder Interfaces.

interface ILogikKonfig

Zuständigkeit

Alle Klassen, die dieses Interface implementieren, sollen für `Logik` eine Funktion `ILogikKonfigDaten Einstellung(String Eigenschaft)` zur Verfügung stellen, die einen Eigenschaftennamen als `String` übergeben bekommt und den zugehörigen Wert als `ILogikKonfigDaten` zurückgibt. So wird beispielsweise die Funktion mit `ATDD` aufgerufen und gibt einen Wert zurück, aus dem ausgelesen werden kann, ob `ATDD` gesetzt ist oder nicht. Auch die Akzeptanztests sind in einer solchen Klasse in einer Liste gespeichert. Diese Liste hat einen Zeiger, der auf den aktuellen Akzeptanztest zeigt. Dieser kann mit den Methoden `ErstenTestAuswaehlen()` und `naechsterTest()` verschoben werden.

Methoden

`public ILogikKonfigDaten Einstellung(String Eigenschaft)`

Diese Methode gibt einen Instanz von `ILogikKonfigDaten` aus der der Wert von der dem Parameter zugeordneten `Eigenschaft` ermittelt werden kann. Falls diese Eigenschaft nicht existiert, wird `null` zurückgegeben.

`public void ErstenTestAuswaehlen()`

Wählt in der Liste der Akzeptanztests den ersten aus.

`public void naechsterTest()`

Wählt in der Liste der Akzeptanztests den nächsten aus.

Abhängigkeiten von anderen Klassen und Interfaces

Das Interface `ILogikKonfig` ist von folgenden Klassen und Interfaces abhängig:

- interface ILogikKonfigDaten

Anhang

Geplante Änderungen

Es soll eine weitere Klasse `TestAuswertung` geben, die `ITestenRueckgabe` in `LogikRueckgabe` konvertiert und dabei die Konsolenrückgabe auswertet. Für die Funktionalität von *BabySteps* müssen noch weitere Funktionalitäten geschaffen werden. Sie könnte in das *Tracking* integriert werden, da hier ebenfalls eine Zeitmessung erfolgt. In diesem Bericht sollen die Klassen besser getrennt werden.

Änderungsprotokoll

21.06.2016	• Das Dokument wurde erstellt
22.06.2015	• Seperation der einzelnen Klassen • Nach dem *Depencency Inversion Principle* (DIP) wurden alle Referenzen zu abhängigen Klassen in Interfaces abgeändert • `LogikRueckgabe` durch `ITestenRueckgabe` ersetzt • Hinzugefügt: `ITestenUebergabe` und interface ILogikKonfigDaten • „Interaktion mit [...]“ in „Abhängigkeiten von [...]“ abgeändert