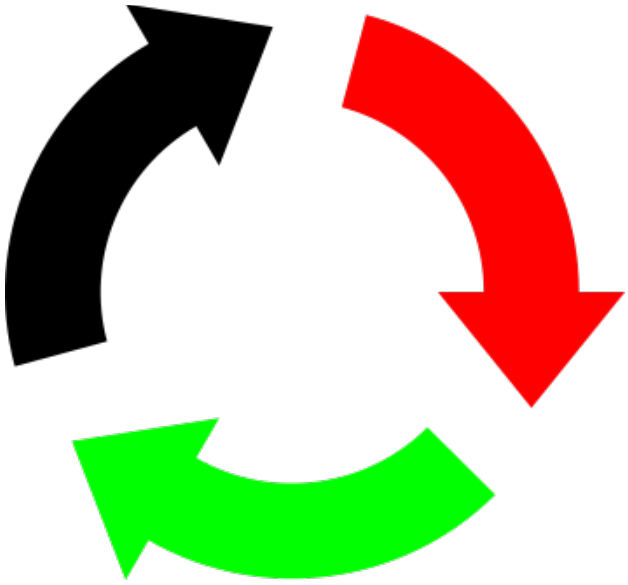
TDDT-Handbuch



Abschlussprojekt des Programmierpraktikums 2016 an der Heinrich-Heine-Universität Düsseldorf

von Let's play ProPra

1. Inhaltsverzeichnis

2. Was ist TDD und was macht dieses Programm?

Test-driven development (TDD) ist eine Methode, um Programme zu entwickeln. Dabei schreibt man abwechselnd Tests und Codes in der folgenden Reihenfolge, bis das Programm fertig ist und das tut, was man möchte:

1. Zunächst schreibt man einen fehlschlagenden Test, der auf eine gewünschte Eigenschaft hin testet.
2. Ändere den Code so, dass der Test nicht mehr fehlschlägt.
3. „Verschönere“ den Code, indem bereits implementierte Dinge sinnvoll zusammengefasst werden.

Dieses Programm ist dazu da, dem Nutzer das TDD nahe zu bringen, indem verschiedene Aufgaben gestellt und mittels des obigen Schemas gelöst werden müssen.

Im Folgenden werden wir näher auf den Programmaufbau eingehen und uns nicht weiter mit TDD befassen, da dies ein Handbuch zum TDD-Trainer (TDDT) sein soll und keine Anleitung für TDD ;)

Wer mehr erfahren möchte, kann auf diversen Internetseiten und Büchern schmökern, z.B. unter https://de.wikipedia.org/wiki/Testgetriebene_Entwicklung

3. Menü-Leiste

Die Menü-Leiste beinhaltet 4 Haupt-Punkte:

- **Menü:** Von dort kann man eine neue Übung wählen oder das Programm beenden
- **Katalog:** Hier wechelt man zwischen verschiedenen Katalogen und Aufgaben. Es besteht die Möglichkeit, selbst erstellte Kataloge zu laden.
- **Tracking:** Dort kann man sich die Phasenanalyse anzeigen lassen.
- **Hilfe:** Hier kommt man zur Aufgabenstellung, falls man sich verrennt oder nicht klar ist, was man dort implementieren soll.

4. Phasen

Wie oben schon beschrieben, besteht das TDD im wesentlichen aus 3 Phasen: Dem *Testschreiben*, dem eigentlichen *Coden* und dem *Refactor*-Schritt.

Test

Am Anfang eines Zyklus steht immer ein fehlschlagender Test. Die rechte Seite des Programms ist die Seite, die zu Beginn aktiv ist und in die die Tests geschrieben werden müssen.

Code

Schlägt genau ein Test fehl, darf man zur Code-Sektion wechseln. Hier schreibt man solange Code, bis der in der vorigen Phase fehlschlagende Test keinen Fehler mehr produziert.

Refactor

Zur Refactor-Phase darf man wechseln, wenn der Code compiliert und kein Test fehlschlägt. Hier kann man (auch bei aktiviertem Babysteps) in Ruhe seinen Code verbessern. Wenn der Code dann immernoch kompiliert werden kann und und auch kein Test verletzt wurde, kann wieder zu Test gewechselt werden und dieser Zyklus-Durchlauf ist beendet.

5. Features

Tracking

Wie oben schon bei der Menü-Beschriebung erwähnt, gibt es eine Phasenanalyse: Sie zeigt dem Nutzer die Zeit, die er in den einzelnen Phasen verbracht hat, sowie die Änderungen, die gemacht wurden, an, um ihm die Möglichkeit zu geben, bestimmte Verhaltensmuster beim Codeschreiben zu erkennen und eventuell zu verbessern.

Babysteps

Es gibt so gennante Babystep-Aufgaben. Diese besitzen eine zeitbeschränkte Code- sowie Testphase, sodass man möglichst kleinschrittig vorgeht. Wird die Zeit nicht eingehalten, so wird in die vorige Phase zurückgewechselt und die bis dato gemachten Änderungen der nicht-bestandenen Phase gelöscht.

6. Sammlung von Fehlerausgaben

TODO