Der

Test Driven Development Trainer

-ein unnötiges Handbuch

Test Driven Development

Das Prinzip von TDD ist relativ einfach zu erklären:

- Du beginnst nicht damit, Code zu schreiben, sondern klatschst erst einmal einen kleinen Test hin, der eine Anforderung an das Programm abdeckt. (Bsp. „Programmiere einen Taschenrechner“. Erster Test: 2+2=4)
- Dann schreibst du minimalen Code, der diesen Test erfüllt – nicht mehr und nicht weniger!
- Dann lehnst du dich gemütlich zurück und betrachtest stolz dein Werk. Fallen dir schon jetzt Wege ein, deine Zeilen schöner zu machen – sei es durch Zusammenlegen von if-Schleifen, Kommentieren von komplizierteren Stellen oder Trennen des Spaghetticodes in kleine Methoden- dann hast du jetzt Zeit dafür. Solltest du nichts finden, beginne wieder bei Schritt 1, bis das Programm genau das tut, was gefordert ist.

Schwieriger wird es vermutlich sein, dich davon zu überzeugen, dass dieses Prozedere sinnvoll ist. Man liest die Aufgabe, hat meistens immer sofort eine grobe Vorstellung von dem, wie man das lösen könnte, und es kribbelt einem in den Fingern, direkt loszulegen – und vor allem will man sich nicht mit Tests schreiben abmühen.

Aaaaaaaber.

Ein weiser Mann sagte einst während einer Vorlesung, dass sich das Schreiben von Tests immer lohnen würde. Und wenn ein weiser Mann das sagt, stimmt das auch. Wir könnten jetzt Argumente bringen von wegen „Man macht sich vorher Gedanken darum, was man tun will“ oder „Dann hat man auch Gewissheit, dass das, was man da schreibt, auch was Sinnvolles tut“ oder „Wenn was fehlschlägt, sieht man sofort, dass man was kaputt gemacht hat“, aber das werden wir nicht tun.

Nimm es einfach hin. Es ist zu deinem eigenen Wohl.

Und weil wir so nett sind *(und weil es ein verpflichtendes Projekt für das ProPra 2016 war)* wollen wir dir das Ganze mit dem hier vorliegenden Programm, dem TDD-Trainer, erleichtern. Wie genau, erfährst du im nächsten Abschnitt.

Der TDD-Trainer: Die Absicht dahinter

Der Kerngedanke hinterm TDDT ist das Verringern des Bürokratieaufwandes auf das minimal Notwendige, damit du dich auf das eigentliche Entwickeln konzentrieren kannst.

Das heißt: kein manuelles Kompilieren (weder von Code noch von Tests – den Befehl für JUnit-compiling konnte sich eh keiner merken) und eine direkte Rückmeldung, ob Code und/oder Test den Anforderungen genügen, damit du in die nächste Phase darfst.

Zusätzlich gibt es zwei **Erweiterungen**:

Die „Babysteps"-Funktion zwingt dich dazu, den Code in winzig kleinen Schritten zu schreiben. Du hast für das Schreiben von Code und Test dann nur jeweils 180 Sekunden (oder die selber eingestellte Zeit) – brauchst du länger, geht dein Code einfach verloren, und du darfst wieder bei der letzten Phase starten. Für die Verbesserungsphase gibt es jedoch kein Limit.

Die „Tracking"-Funktion schaut dir immer über die Schulter und schreibt mit, was du wann verzapft hast, damit du am Ende einen Bericht hast, den du dir dann ausdrucken und übers Bett hängen kannst. Du könntest ihn auch dazu verwenden, um herauszufinden, wo du häufig viel Zeit verschwendest, aber hey – lass dir von uns nicht sagen, was du tun sollst.

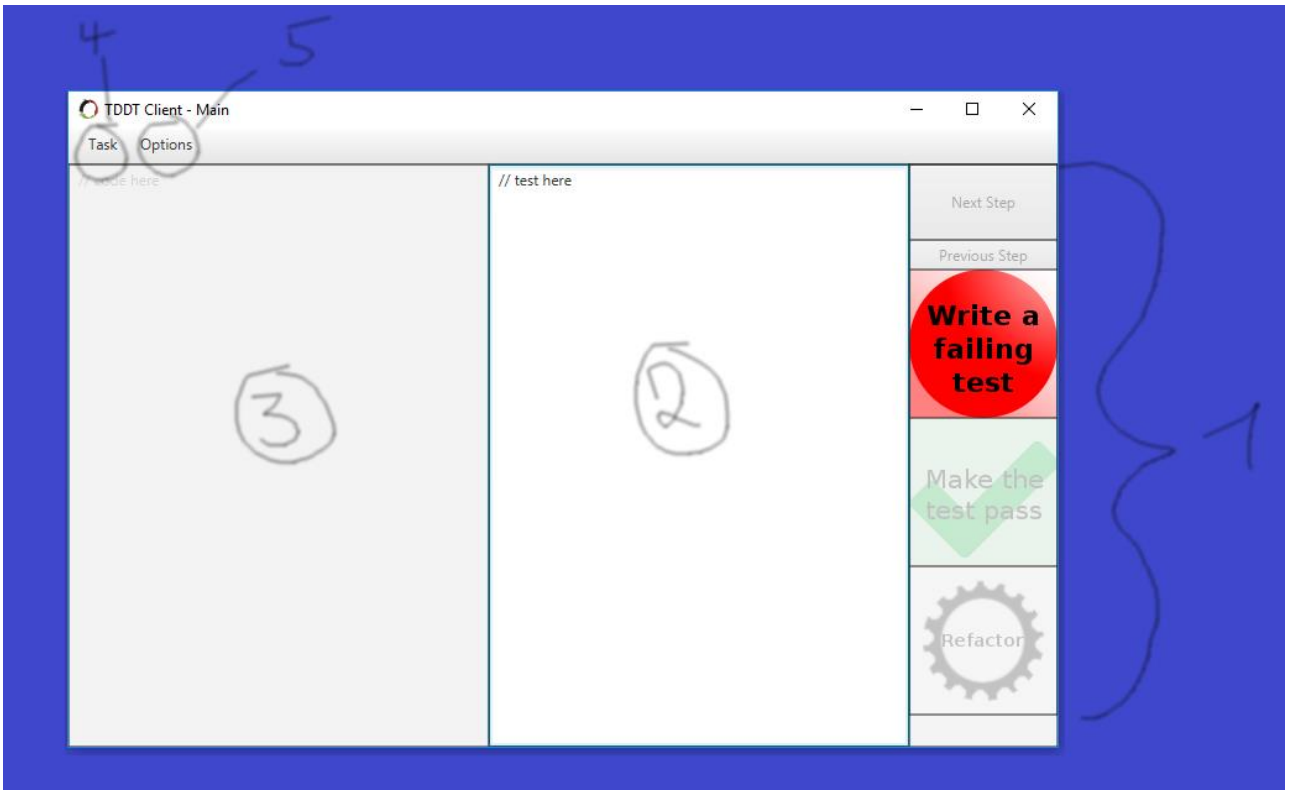
Übersicht: Erstes Benutzen des Programmes

Nutzt du das Programm zum ersten Mal, sieht der vorgesehene Ablauf wie folgt aus:

1) Task -> New Excercise
   Hier erstellst du eine neue Aufgabe.
2) -> Next
   Jetzt musst du ein Grundgerüst für die Testklasse und die Mainklasse angeben.
3) -> Submit
   Wähle „Create a new catalog" aus und merke dir den Speicherort.
4) Task -> Open Catalog -> File -> Load Catalog
   Wähle den gerade erstellten Katalog aus.
5) Nun kannst du in der linken Spalte des Excercise Choosers die erstellte Aufgabe auswählen. Rechts siehst du nochmal einen Überblick über die Einstellungen. Klicke „Choose"
6) In den Textfeldern sollten nun die Startskelette der zwei Klassen stehen und du kannst mit dem Programmieren loslegen!

Das GUI

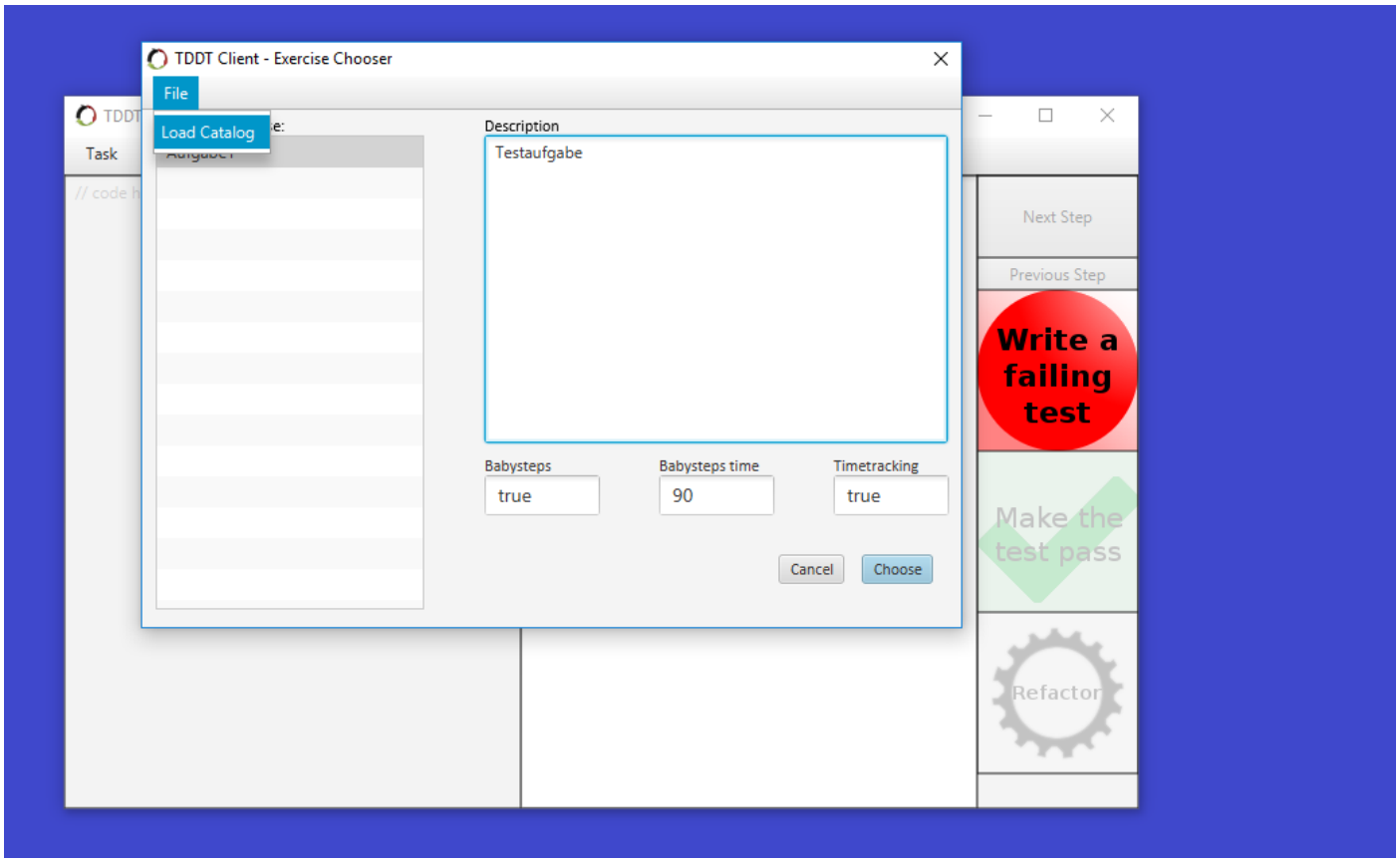
1. Überblick



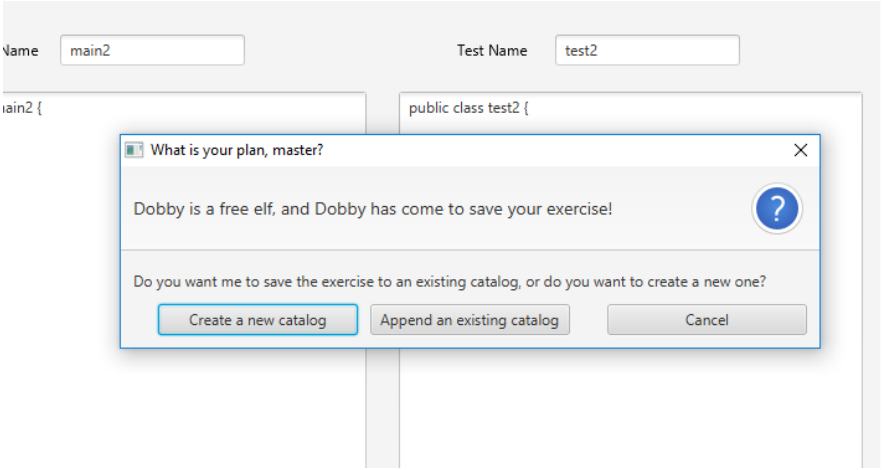
1) Hier siehst du, in welcher Phase du momentan steckst, und kannst versuchen, in die nächste Phase zu gelangen
2) Hier kommen deine Tests rein
3) Hier kommt dein Code rein
4) Hier kannst du Aufgaben und den Katalog verwalten
5) Hier kommst du zu den Optionen für Tracker und Babysteps

2. Katalog: Anlegen und Auswählen von Aufgaben

Im Task-Dropdown gibt es zwei Optionen:
Unter **Open Catalog** wird der aktuelle Katalog mit allen darin enthaltenen Aufgaben angezeigt. Zusätzlich finden sich hier Infos zu den default-Einstellungen von Babysteps und Tracking.
Hier kannst du eine Aufgabe auswählen, die du bearbeiten möchtest, oder einen bereits vorhandenen Katalog einlesen:



Der Katalog besteht aus einer .xml-Datei, die automatisch beim Erstellen einer Aufgabe mit **New excercise** angelegt werden kann. Willst du eine neue Aufgabe anlegen, wähle Task -> New Excercise -> Eingabe der Daten -> Next -> Eingabe von Text/Codeskelett. Nun sollte Dobby auf dich zukommen:

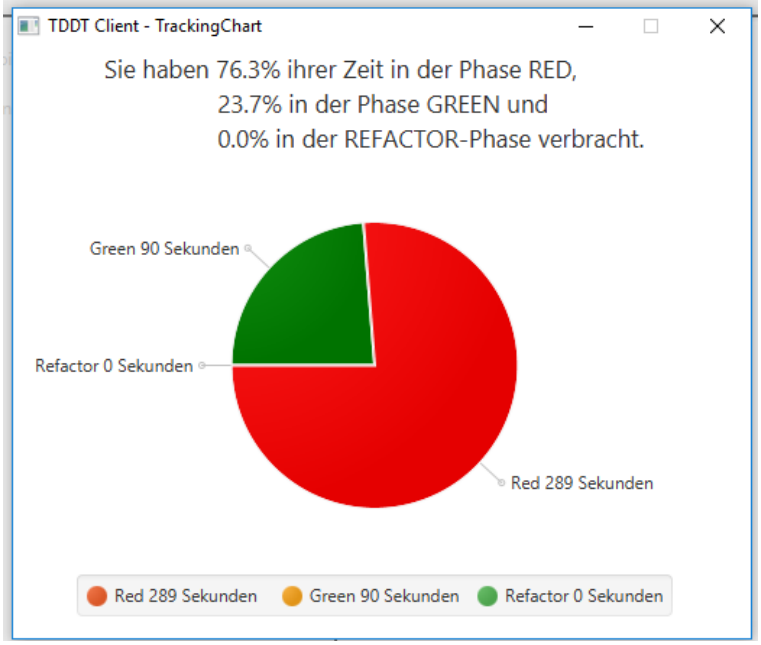


Ist das deine erste Aufgabe, fällt die Entscheidung leicht: Wähle „Create a new catalog“ aus. Nun kannst du den Namen des Kataloges und den Speicherort auswählen. Jetzt wird eine .xml-Datei angelegt, die du nun mit Task -> Open Catalog -> Load Catalog laden kannst.

Willst du die Aufgabe in einen bereits vorhandenen Katalog einpflegen, wähle „Append an existing catalog“ und gebe die .xml-Datei an.

3. Zusätzliche Optionen

Das Menüitem „Options“ gibt dir Zugriff auf die Tracker-Erweiterung des Programmes. Hier kannst du dir mithilfe der Trackerfunktion ein Chart anzeigen lassen, wo du wie viel Zeit verwendet hast:



(So sollte ein Chart besser nicht aussehen)

Willst du nachvollziehen, wann du welche Änderungen gemacht hast, gehe auf Options -> Tracker -> Show all, um eine History aller Änderungen zu bekommen, die du in der aktuellen Sitzung ausgeführt hast. Sogar mit Uhrzeit.