Projekt 7
TDDT

# Handbuch

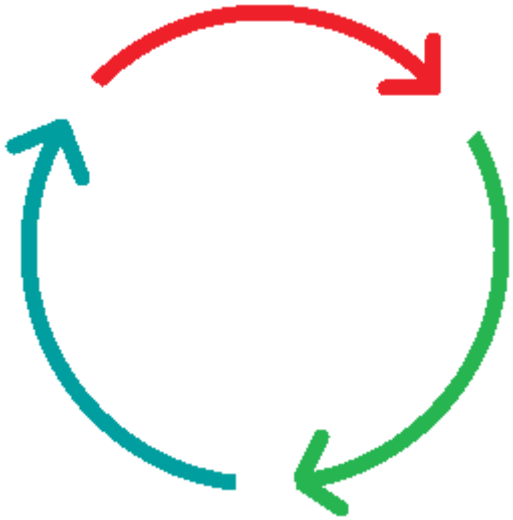
Gruppe: Jünger des Binärius
Marius May
Nils Kleine Klausing
Can Schwarzkamp

*Stand: 14.07.2016*

# Einleitung

## TDDT - TestDrivenDevelopmentTrainer

TDD – TestDrivenDevelopment *(dt. testgetriebene Entwicklung)* bezeichnet eine Arbeitsweise in der Programmierung, bei der zunächst ein Test geschrieben wird bevor der eigentliche Programmtext verfasst wird. Man arbeitet in 3 Phasen (Statusse):



### RED

In diesem Status wird ein fehlschlagender Test verfasst. RED ist der Start des Zyklusses.

### GREEN

In diesem Status wird der Programmtext verfasst, sodass der vorher geschriebene Test erfolgreich ist. Man erreicht diesen Status, wenn im Status RED genau ein Test fehlschlägt oder der Code nicht kompiliert werden kann. Alternativ kann man wieder zu RED zurückkehren, dabei wird allerdings der Fortschritt in RED und GREEN gelöscht und der Zyklus beginnt von neu.

### REFACTOR

In diesem Status kann der geschriebene Code verbessert werden. Man erreicht diesen Status nur wenn im Status GREEN alles fehlerfrei kompiliert und alle Tests erfolgreich sind und kann ihn auch nur in diesem Fall wieder verlassen um wieder in den Status RED zu gelangen.

Weitere Informationen unter http://www.frankwestphal.de/TestgetriebeneEntwicklung.html.

# Bedienung des Programms

Mithilfe des Programms können testgetrieben nach dem oben erklärten Schema verschiedene Aufgaben aus einem Katalog gelöst werden. Die Installation geht aus der `Readme.md` in unseren GIT-Repository hervor.

## Exercise auswählen

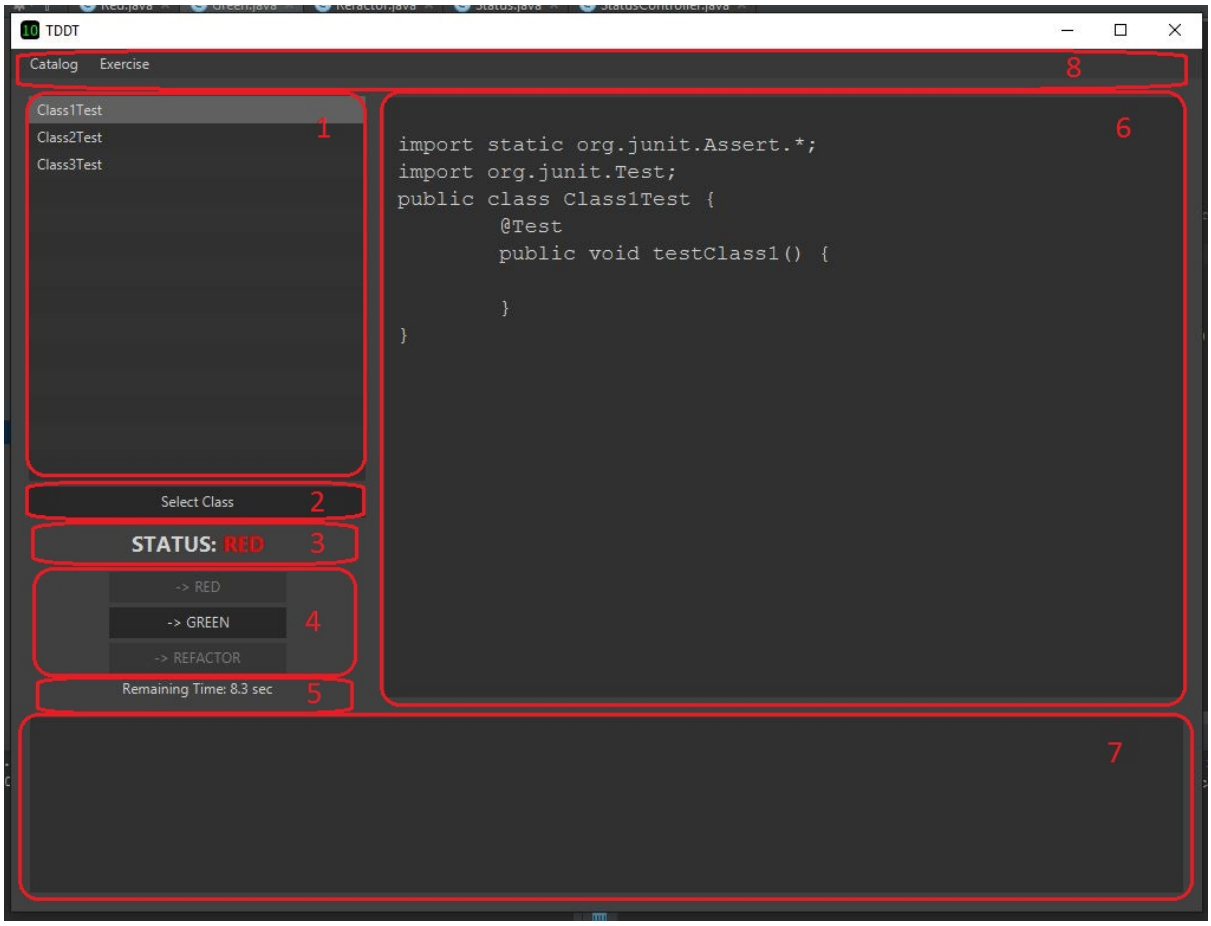
In diesem Fenster wählt man eine Exercise aus der `Catalog.xml` aus. Die Exercises werden von dem Programm eingelesen und verarbeitet.


1. ListView um Exercise auszuwählen. Steht hinter der Exercise “(B)” so ist Babysteps (Erklärung siehe unter Erweiterungen) für die Exercise aktiviert. Ist Babysteps nicht aktiviert, so läuft im Programm dennoch die Zeit runter, sodass dies als Orientierung dient.

2. Button um zu der jeweiligen Exercise ins Hauptfenster zu gelangen.

# Hauptfenster

In dem Hauptfenster wird die Programmierarbeit verrichtet.




1. In dieser Liste werden die Klassen/Tests der Exercise angezeigt, die man verändern kann.

2. Der Button stellt die aktuell selektierte Klasse dar, sodass diese bearbeitet werden kann.

3. Aktueller Status.

4. Die Buttons überführen in den neuen Status. Dabei gilt:
   RED → GREEN
   GREEN → REFACTOR und GREEN → RED
   REFACTOR → RED

5. Verbleibende Zeit von Babysteps.

6. Textfeld zur Darstellung und Bearbeitung des Codes.

7. Textfeld, in welchem Meldungen für den User angezeigt werden.

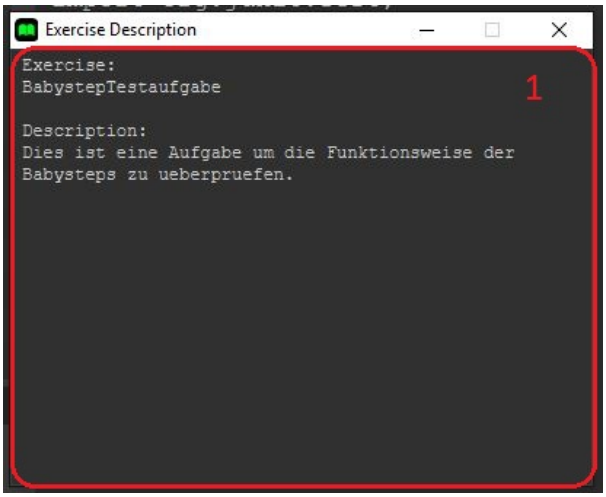
8. Menü-Leiste, in welcher verschiedene Optionen zur Verfügung stehen:

**Catalog**

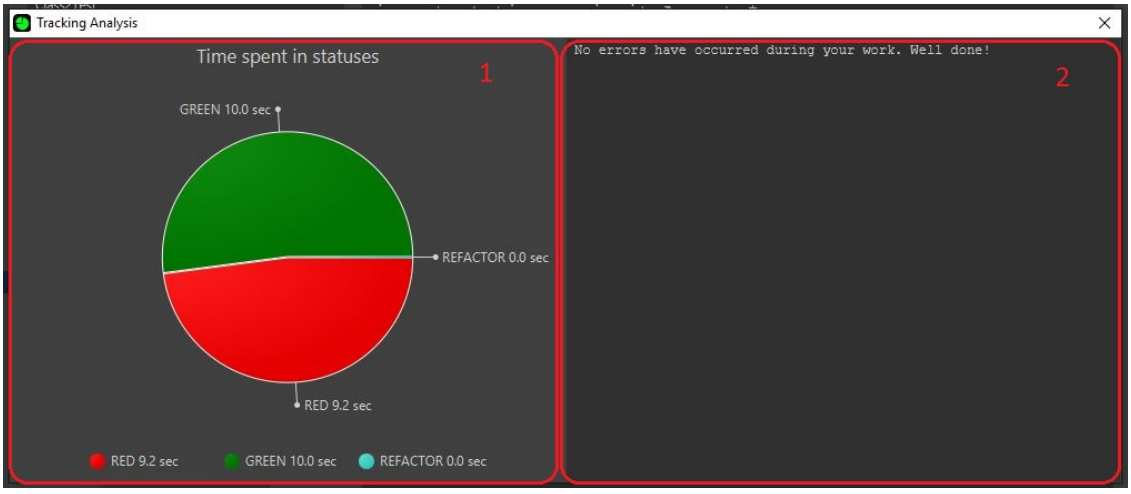
- Load Exercise
  Auswahl einer neuen Exercise.

**Exercise**

- Show Description
  Ausgabe des Exercise-Namen und der Description der Exercise (Aufgabenstellung).




- Show Tracking
  In dem linken Kuchendiagramm wird die Zeit in den einzelnen Phasen dargestellt. In dem rechten Textfeld werden die aufgetretenen Kompilierungsfehler aufgelistet



# Aufbau des Aufgabenkatalogs (XML-File)

`<exercises>`
Hier werden die einzelnen Exercises reingeschrieben.

> `<exercise name="...">`
> Dies beschreibt eine einzelne Exercise, wobei in `name` der Name der Exercise angegeben wird.
>
> > `<description>`Hier wird die Aufgabenstellung abgespeichert.`</description>`
> >
> > `<classes>`
> > Hier werden die Klassen reingeschrieben.
> >
> > > `<class name="...">`
> > > Hier wird eine einzelne Klasse definiert. Zwischen die Tags kommt der Programmcode (Lösungsskelett) und in `name` kommt der Klassenname. Dabei sollte der Name der Klasse idealerweise dem Namen der Klasse im Lösungsskelett entsprechen.
> > > `</class>`
> >
> > `</classes>`
> >
> > `<tests>`
> > Äquivalent zu `<classes>`, aber für Testklassen.
> >
> > > `<test name="...">`
> > > Äquivalent zu `<class>`, aber für Testklassen. Zu beachten ist, dass für jede Testklasse mindestens eine Methode (mit @Test) vordefiniert werden muss, um ungewollte Kompilierungsfehler zu vermeiden.
> > > `</test>`
> >
> > `</tests>`
> >
> > `<config>`
> > Hier kommen die Konfigurationen zu den einzelnen Exercises rein.
> >
> > > `<babysteps statusSwitch="..." time="..." />`
> > > Hier wird die BabySteps-Konfiguration angegeben. In `statusSwitch` gibt man `true` oder `false` an, ob bei abgelaufener Zeit in den vorherigen Status gewechselt werden soll. In `time` wird die Zeit für die BabySteps in Sekunden angegeben.
> >
> > `</config>`
>
> `</exercise>`

`</exercises>`

Den mitgelieferten Aufgabenkatalog können Sie in der `Catalog.xml` einsehen.

# Erweiterungen des Programms

Das Werkzeug besitzt zwei Erweiterungen, welche nun erklärt werden sollen.

## Babysteps

Die Phasen RED und GREEN werden zeitlich begrenzt. In Status RED wird bei Ablauf der Zeit der neu geschriebene Code gelöscht und der Timer beginnt erneut. Läuft hingegen im Status GREEN die Zeit ab, so wird nicht nur der neu verfasste Code gelöscht, sondern es wird zusätzlich in den vorherigen Status RED gewechselt, wo ebenfalls der Fortschritt verfällt, man befindet sich also wieder am Anfang des Zyklusses und die Zeit läuft erneut runter.

Ob Babysteps eingeschaltet ist und welche Countdown-Zeit eine Aufgabe hat, entscheidet der entsprechende Eintrag im Aufgabenkatalog. Es ist zu beachten, dass auch bei ausgeschalteten Babysteps die Zeit trotzdem abläuft und angezeigt wird, sodass man während des Programmierens ein Gefühl für die verlangten Zeitlimits bekommt.

## Tracking

Bei Tracking handelt es sich um eine Analyse der erledigten Arbeit. Aufgerufen über die Menü-Leiste kann einem jederzeit eine genaue Auswertung angezeigt werden. Dazu gehört ein Diagramm, welches die Dauer angibt, die man in den verschiedenen Statussen verbracht hat, sowie eine Auflistung der aufgetretenen Kompilierungsfehler.