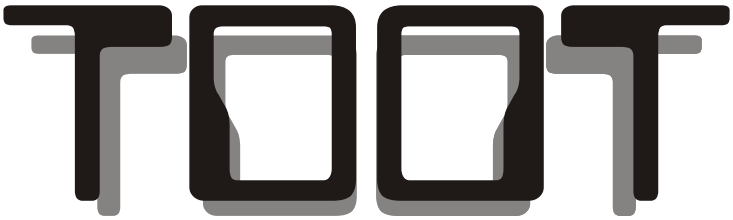


Test Driven Development Trainer

von den Ävaders

Nutzerhandbuch

Inhaltsverzeichnis

1. Installation

1.1 Systemvoraussetzungen

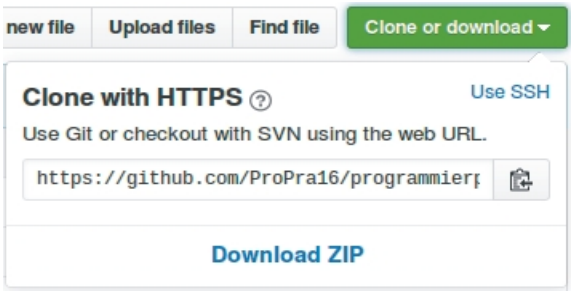
TDDT ist in Java geschrieben und daher plattformunabhängig. Das Programm kann auf jedem Rechner ausgeführt werden, auf dem eine JVM (Java Virtual Machine) installiert ist.

1.2 Herunterladen

TDDT ist als Projekt auf der Entwicklerplattform `github` verfügbar. Um das Projekt herunterzuladen rufen sie zuerst die Projektseite auf `github` auf:

https://github.com/ProPra16/programmierpraktikum-abschlussprojekt-the-avaders

Nun klicken sie auf `Clone or download` und wählen `Download ZIP`.



Das aufgerufene ZIP-Archiv speichern sie auf ihrem Rechner und entpacken es in ein seperates Verzeichnis, im folgenden als „Hauptverzeichnis“ bezeichnet.

1.3 Kompilieren und ausführen

Im Hauptverzeichnis sind die Dateien `gradlew` und `gradlew.bat` abgelegt, über die sich gradle starten lässt. Das folgende Vorgehen ist abhängig von ihrem Betriebssystem:

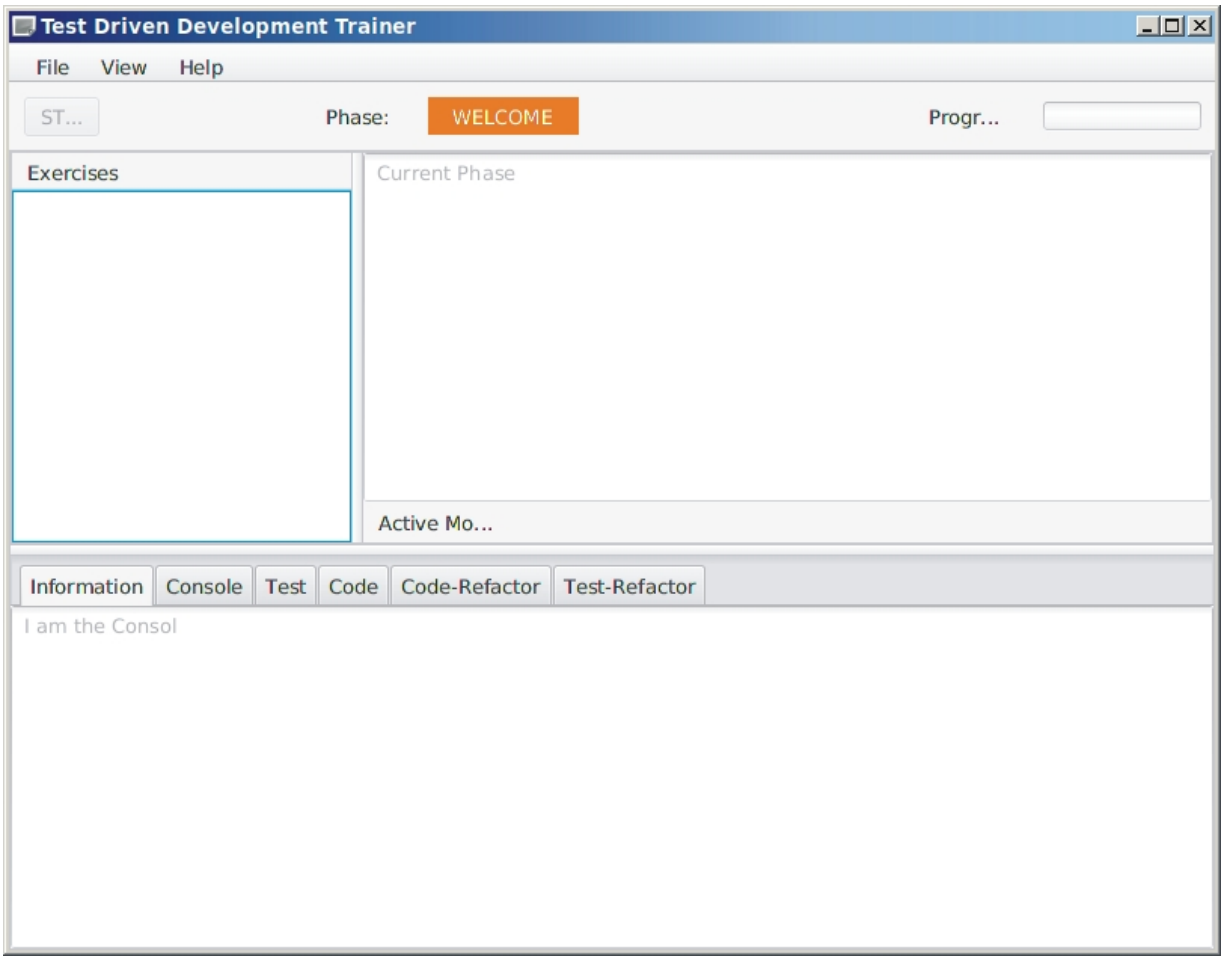
Unter Windows öffnen sie ein Kommandozeilenfenster und wechseln in das Hauptverzeichnis. Dann führen sie `gradlew.bat` aus.

Auf Unix-Systemen öffnen sie ein Konsolenfenster im TDDT-Verzeichnis und starten das Programm mit `./gradlew` dirkt.

Die bloße Ausführung bewirkt nur, dass Gradle ordnungsgemäß installiert wird. Zum starten des Programms muss doch ein Parameter angehangen werden. Der Aufruf lautet dann `gradlew.bat run` bzw. `./gradlew run`. Diese Anweisung lässt Gradle weitere Dateien herunterladen, die zur Ausführung des Programms notwendig sind. Anschließend wird das Programm kompiliert und gestartet.

2. Verwendung

2.1 Hauptfenster



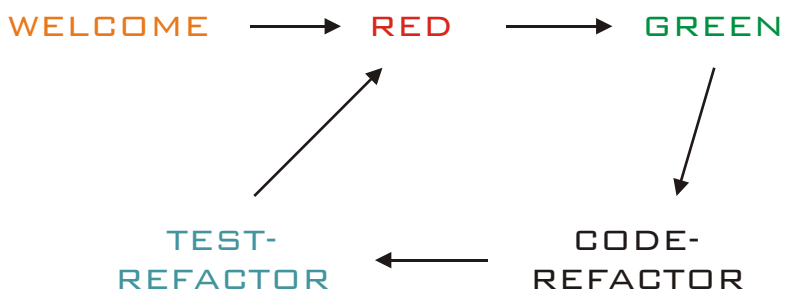
Das Hauptfenster von TDDT besteht aus vielen Komponenten. Mittig befindet sich das Textfeld für den eigentlichen Quelltext, hier mit „Current Phase“ beschriftet. Alle Eingaben des Nutzers laufen über dieses Textfeld.
Der Informationsabschnitt darunter versorgt den Nutzer dagegen mit Daten. In einzelnen Registerkarten wird automatisch auf die für den Nutzer relevante gewechselt. Diese werden jedoch im folgenden noch genauer beschrieben.

Auf der linken Seite befindet sich eine Baumansicht. Hier können zunächst Aufgaben aus dem Katalog ausgewählt werden. Mit dem nebenstehenden Pfeil kann die Aufgabe ausgeklappt werden. Sie enthält zwei Unterpunkte `Classes` und `Test`. Während des Entwicklungszyklus werden sie immer zwischen einzelnen Quelltexten in `Classes` und `Test` wechseln. Durch das einklicken eines Quelltextes wird dieser in das bereits erwähnte Textfeld geladen.
Der aktuelle Entwicklungsschritt wird als Phase in der oberen Leiste angezeigt. Daneben befindet sich eine Fortschrittsanzeige. Beide werden später noch genauer betrachtet.

2.2 Laden einer Aufgabe

Zuerst wird über die Menüleiste ganz oben eine neue Konfigurationsdatei geladen. Dazu wird der Punkt `File` → `New Catalogue` ausgewählt. Es erscheint ein Dialogfenster, in dem sie eine entsprechende Datei auswählen können. Diese Datei hat die Endung `*.xml` und muss der Spezifikation entsprechen.
Ist die Datei fehlerhaft, wird eine entsprechende Fehlermeldung im Reiter `Information` angezeigt. Andernfalls können dort der Name der Aufgabe und eine Aufgabenbeschreibung abgelesen werden. In die Baumansicht wurden alle verfügbaren Aufgaben aus der Datei geladen.

2.3 Phasenmodell



Die testgetriebene Entwicklung umfasst mehrere Entwicklungsschritte, auch als Phasen bezeichnet. Durch die strikte Beachtung der Regeln für testgetriebenes entwickeln sollen möglichst fehlerfreie Programme geschrieben werden. Dazu wird jede Funktion im eigentlichen Quelltext durch einen Test abgesichert. Dieser ist selbst ein kleines Programm, das den Quelltext überprüfen soll.
Dazu ruft es die zu prüfende Funktion mit konkreten Werten auf und prüft die Rückgabewerte. Soll eine Funktion bespielsweise das Quadrat einer Zahl berechnen, ruft das Testprogramm sie mit einigen Zahlen auf und vergleicht den Rückgabewert mit dem geforderten Ergebnis.

2.3.1 RED

Die erste Phase im Entwicklungsschritt wird in TDDT als RED bezeichnet. Wählen sie zuerst in der Baumansicht eine Testklasse aus. Starten sie nun die erste Phase, indem sie oben links auf START klicken. Je nach verwendeten Erweiterungen startet nun auch eine Stoppuhr.
Sie können nun die Testklasse editieren. Dabei sollen sie nun genau ein weitere Testfunktion schreiben, der fehlschlägt. Durch einen Klick auf NEXT oben rechts können sie ihr Programm kompilieren lassen. Der Informationabschnitt wechselt auf den Reiter Console und zeigt ihnen die Ergebnisse des Compilers an. Falls ein Compiler-Fehler auftritt, wird dieser ebenfalls in diesem Reiter angezeigt.

2.3.2 GREEN

Nachdem sie nun einen fehlschlagenden Test geschrieben haben, gilt es, dies zu beheben. Dazu wird das Programm so weit erweitert, dass der Test durchläuft. Dabei soll darauf geachtet werden, dass der Quelltext nicht mehr als nötig erweitert wird. Bei kleinschrittigem Vorgehen und kleinen Aufgaben reicht das Hinzufügen oder Ändern weniger Zeilen bereits aus. Der Test wird dabei im Informationsabschnitt angezeigt.
Oben rechts können sie durch einen weiteren Klick auf NEXT das Programm kompilieren lassen. Falls Fehler auftreten, werden diese im Konsolenreiter angezeigt. Andernfalls wird zum nächsten Entwicklungsschritt gewechselt. Falls sie es nicht schaffen, den Test zu erfüllen, können sie auch durch einen Klick auf BACK zu RED zurückkehren. Dabei werden jedoch alle vorgenommenen Änderungen am Quelltext in GREEN verworfen.

2.3.3 REFACTOR-CODE

Nach der eigentlichen Weiterentwicklung des Programms folgt nun das aufräumen. Da bei der Entwicklung möglichst wenige Änderungen vorgenommen werdern sollen, entstehen oft Strukturen, die sich zusammenfassen lassen. Beispielsweise können Anweisungen zu Schleifen zusammengefasst oder wiederkehrende Befehlsgruppen in Funktionen ausgelagert werden. Auch eine Gruppe aus `if`-Bedingungen kann gegebenenfalls zu einem Abschnitt mit `switch` zusammengefasst werden. Nach Abschluss dieser Phase wird das Programm wieder kompiliert. Falls dabei kein Fehler auftritt und alle Tests durchlaufen, wird zur nächsten Phase weitergegangen. Andernfalls können sie über den Reiter Console die Fehler ablesen. Im Reiter Code können sie den alten, getesteten Quelltext ablesen und auch gegebenfalls in das Textfeld kopieren, um die Phase neu zu starten.

2.3.4 REFACTOR-TEST

Ähnlich der vorherigen Phase haben sie nun die Mglichkeit, den Test aufzuräumen, indem sie gegenbenenfalls wieder Strukturen zusammenfassen oder Befehlsgruppen in Funktionen auslagern. Nach Abschluss dieser Phase wird das Programm wieder kompiliert und alle Tests müssen durchlaufen, bevor zum nächsten Entwicklungsschritt gewechselt werden kann.

Nach Abschluss dieser Phase beginnt der Entwicklungszyklus von neuem mit `RED`. Sie müssen also einen weiteren fehlschlagenden Test schreiben. Dabei darf jedoch kein anderer Test fehlschlagen, es muss also immer genau ein Test fehlschlagen. Falls es dabei zu Problemen kommen sollte, können sie den alten Test aus dem Reiter `Test` in das Textfeld kopieren. Sie können auch den gesamten Entwicklungszyklus beenden, indem sie in der Menüleiste `File` → `End Cycle` auswählen. Dabei werden die Änderungen des letzten abgeschlossenen Zyklus gespeichert.
Falls die also der Meinung sein sollten, mit der Entwicklung fertig zu sein, wechseln sie nach `REFACTOR-TEST` zu `RED` und beenden den Entwicklungszyklus.

2.4 Erweiterungen

Zur Steigerung des Lerneffekts und damit möglicherweise auch des Schwierigkeitsgrades für den ungeübten Nutzer wird die Zeit begrenzt, die dem Nutzer zum editieren des Quelltextes zur Verfügung steht. Diese Erweiterung ist an einem Eintrag wie

$$\rightarrow \text{Babysteps},\ 120\text{sec}$$

erkennbar. In diesem Beispiel stehen dem Nutzer 120 Sekunden zur Verfügung, bevor die Phase zurückgesetzt wird.

3. Referenzen

3.1 **TDDT-Repository**
Das Repository auf GitHub, aus dem sie sich TDDT herunterladen können
https://github.com/ProPra16/programmierpraktikum-abschlussprojekt-the-avaders

3.2 **JVM (Java Virtual Machine)**
Download auf der Webseite von Oracle
http://www.java.com/de/download/manual.jsp

3.3 **Spezifikation für Konfiguartionsdateien**
Anleitung zum Erstellen von Konfiguartionsdateien
https://github.com/ProPra16/programmierpraktikum-abschlussprojekt-the-avaders/blob/master/doc/User%20Guide%20-%20Exercise%20catalogues%20in%20XML.pdf

3.4 **Testgetriebene Entwicklung (engl. TDD)**
Text von Frank Westphal
http://www.frankwestphal.de/TestgetriebeneEntwicklung.html

3.5 **Dokumentation**
Beschreibung des Programms mit seinen Klassen und Funktionen
https://github.com/ProPra16/programmierpraktikum-abschlussprojekt-the-avaders/blob/master/doc/

3.6 **Bildquellen**
`Download ZIP` - Screenshot von github, Referenz unter 3.1 angegeben
`Hauptfenster` - Screenshot von TDDT, Referenz unter 1.2 angegeben
Soweit nicht anders gekennzeichnet stammt das Material vom Herausgeber

@Florian-Dt