Test Driven Development Trainer

Benutzerhandbuch

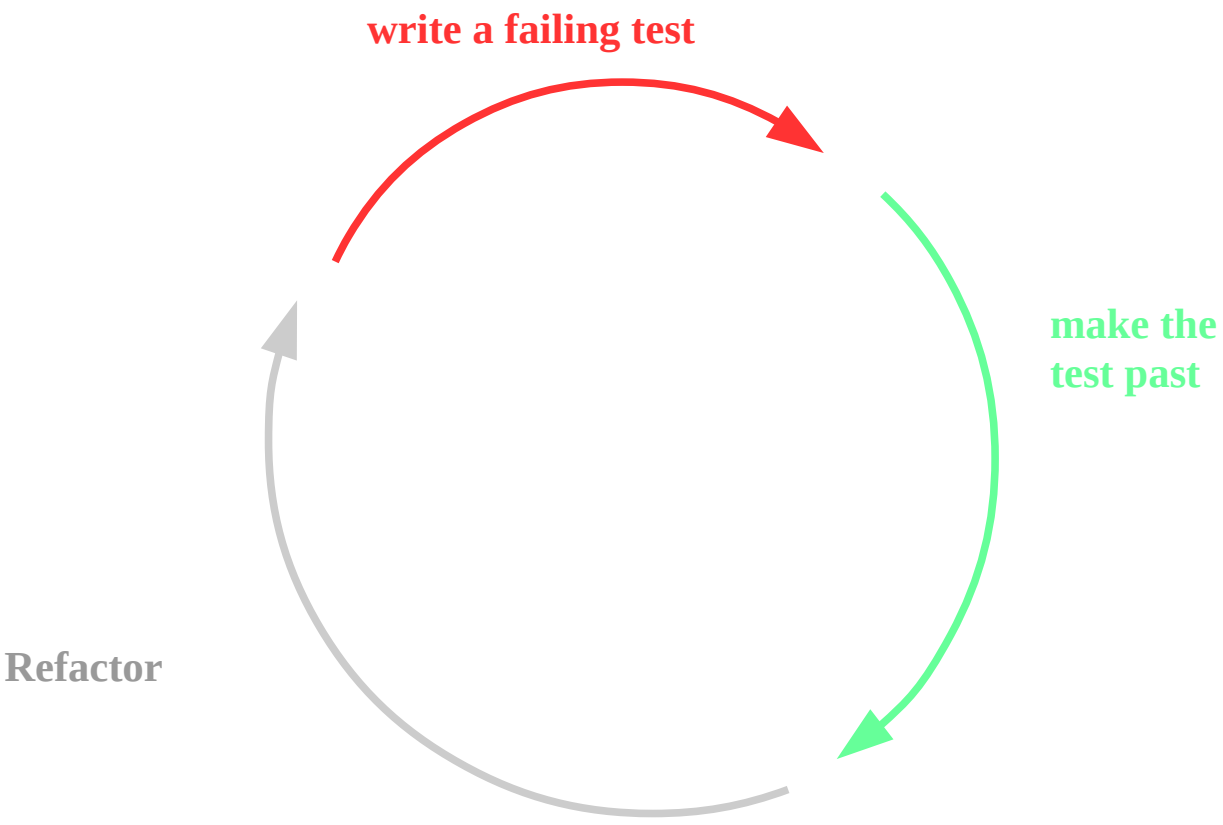
Andreas Burbach, Antonio Berycka, Olexiy Chornovil, Stefan Kerren

Inhaltsübersicht

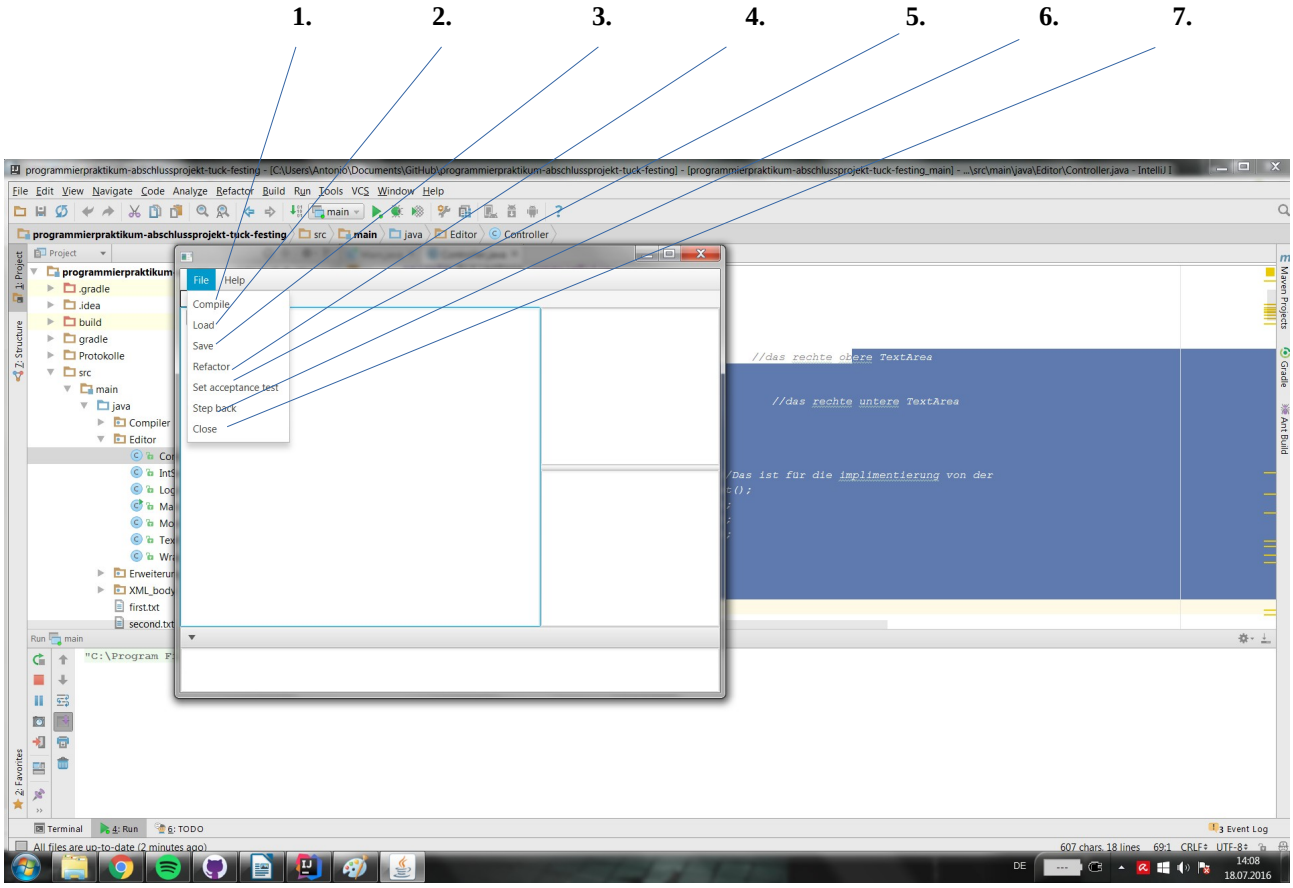
- **Grundidee**……...……2
- **Editor**……...3-4
- **RED**……..5
- **GREEN**…….…6
- **Refactor**…….……7
- **Schritt-für-Schritt Anleitung**……...8

Grundidee

Die Anwendung TDDT soll dazu dienen Studenten in den ersten Semestern zu helfen,
die Technik der testgetriebenen Entwicklung zu üben.
Bei der testgetriebenen Entwicklung handelt es sich um eine Technik zur Softwareentwicklung,
bei der ein Test vor dem zu testenden Code geschrieben wird.
TDDT soll den Prozess abbilden und den Benutzer führen.



Editor



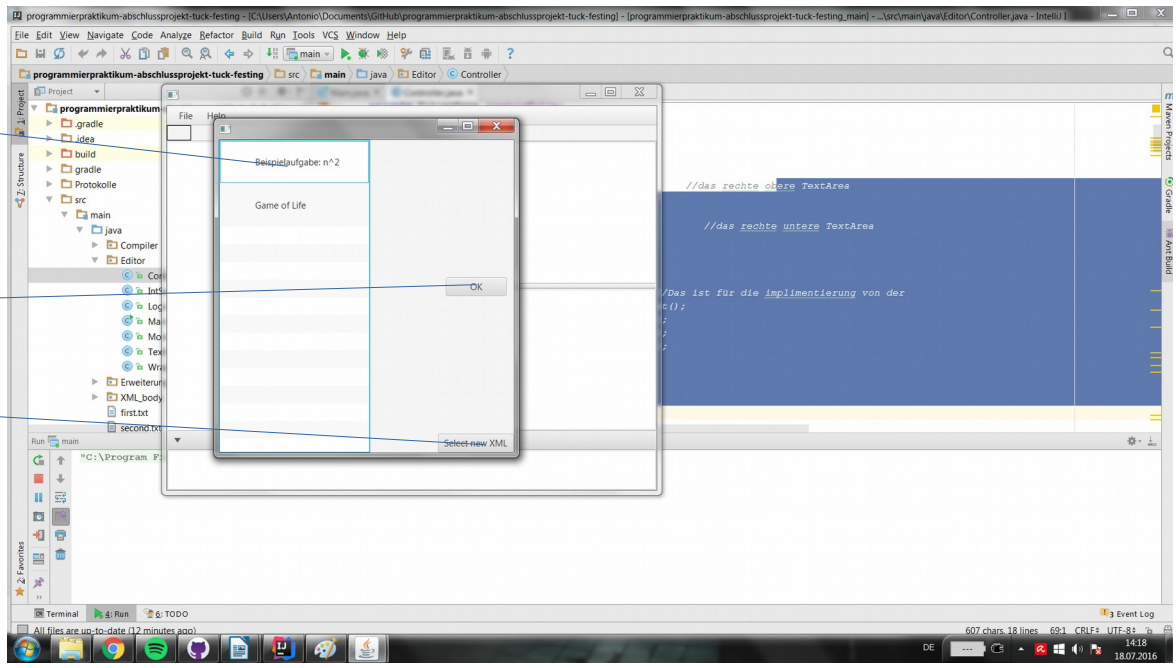
1. **Compile:** Compiliert den geschriebenen Code und schickt den User, je nach dem Code zu dem, dazugehörigem, nächsten Schritt.

2. **Load:** Öffnet das Fenster für die Auswahl der Aufgaben und bietet die Möglichkeit eine, schon gespeicherte Aufgabe auf den Editor zu laden.

Aufgabe

Bestätigung der Aufgabenauswahl

Hinzufügen einer neuen Aufgabe



3. Save: Speichert den, im Editor vorhandenen, Code als XML-Datei.

4. Refactor : Compiliert das Programm und gibt Fehler aus, ohne zu den Tests zu springen.

5. Step Back: Schickt den User zu dem Schritt, in dem er sich vor dem aktuellen Schritt befand

6. Set acceptance test: Öffnet das Fenster, der Akzeptanztests.

7. Close: Beendet das Programm

RED

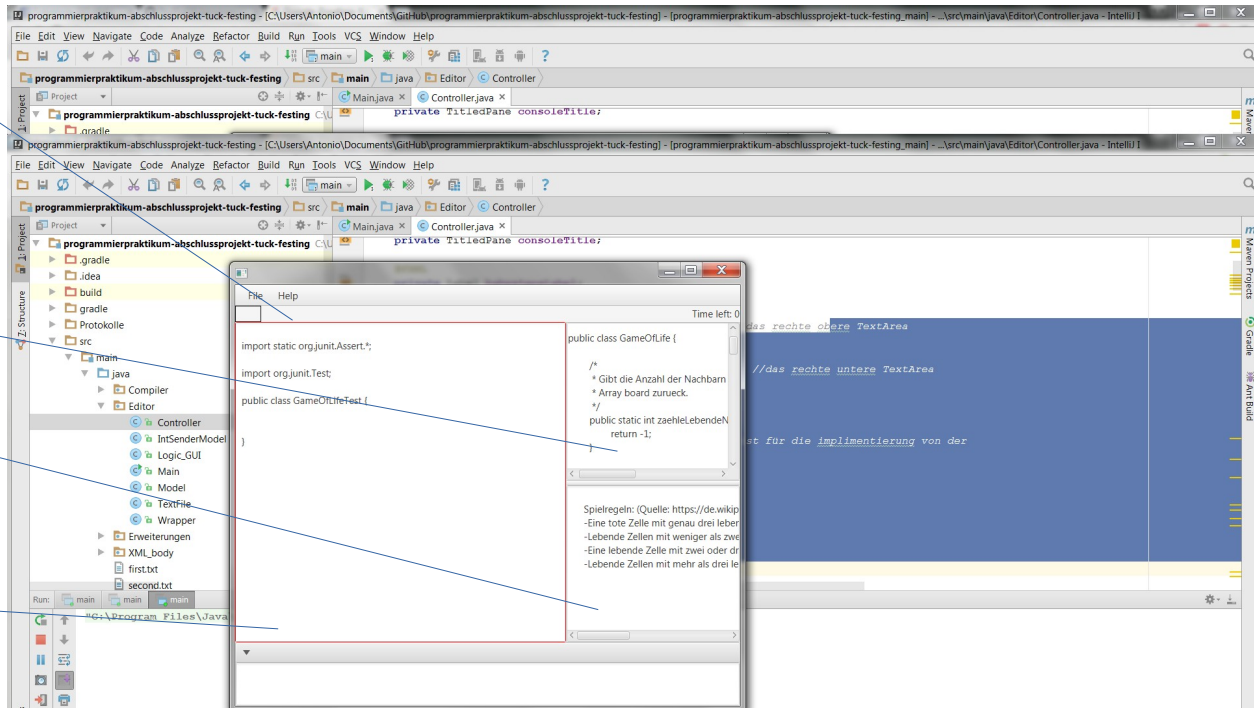
Das Werkzeug erlaubt es dem Nutzer nur den Test zu editieren, bis es genau einen fehlschlagenden Test gibt oder der Code nicht compiliert.
Der Nutzer soll dem Programm mitteilen, dass er bereit ist für den nächsten Schritt.
Der Wechsel zum nächsten Schritt darf aber nur erfolgen, wenn die Bedingung erfüllt ist.

Modusanzeige (Step Red)

Programm-code

Aufgaben-stellung

Testcode



GREEN

Das Programm erlaubt es dem Nutzer nun ausschließlich den Code zu modifizieren, bis alle Tests laufen.
Alternativ kann der Benutzer zurück zu **RED** wechseln .
Beim Zurückwechseln zu den Tests wird der neue Code gelöscht (d.h. der Code befindet sich dann in dem selben Zustand in dem er war, als der Benutzer in den Code-Editier-Modus eingetreten ist).

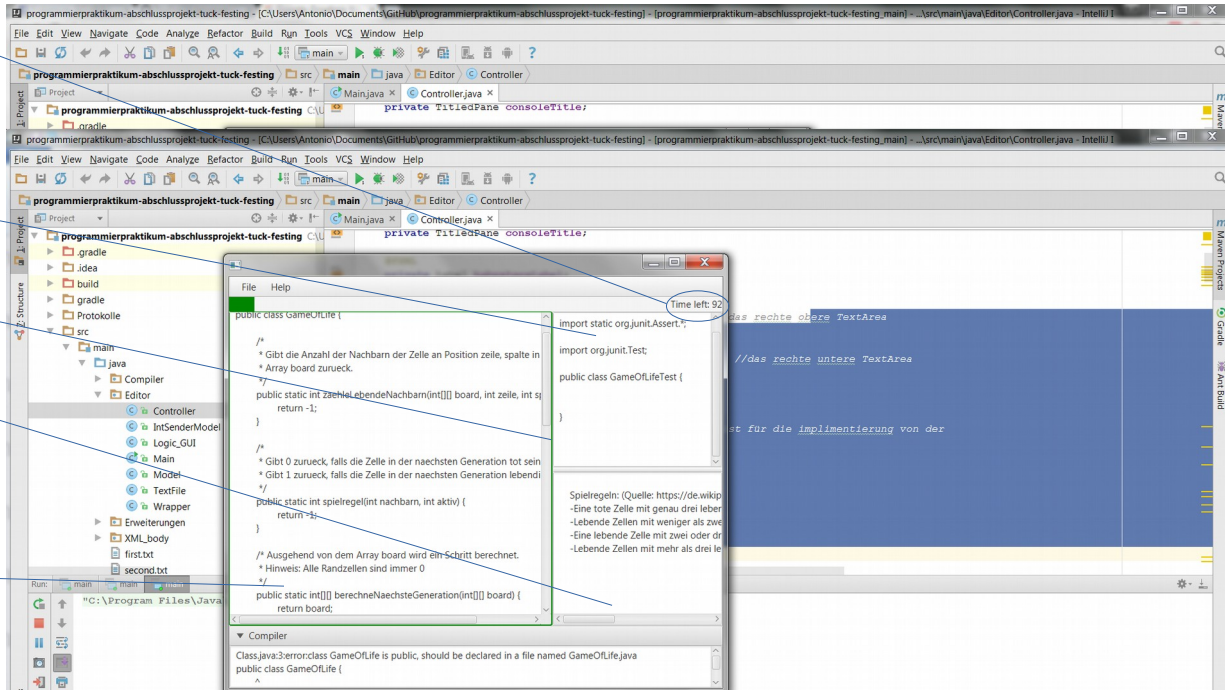
Erweiterung: Babysteps

Testcode

Modusanzeige (Step Green)

Aufgabenstellung

Programmcode



Refactor

Nachdem alle Tests laufen, darf der Nutzer den Code verbessern (refactoring).
Der Wechsel von **GREEN** nach **REFACTOR** muss der Benutzer dem Programm explizit mitteilen.
Es ist nur erlaubt zu wechseln, wenn die Bedingungen
(alles kompiliert und alle Tests laufen durch) erfüllt sind.
Sie können auch einen zweiten Refactor Zustand einfügen:
Nachdem der Code verbessert wurde, können die Tests ebenfalls verbessert werden.
Wichtig ist aber das Refactoring immer bedeutet,
dass alle Tests immer vorher und nachher laufen.

Zusätzliche Schritt-für-Schritt Anleitung zur Bedienung des Programms:

1. Zuerst wählt der Nutzer aus dem Katalog eine Aufgabe mithilfe von 'Load' aus, nach der Auswahl muss noch einmal mit 'OK' bestätigt werden.

2. Jetzt sollte der Nutzer sich im Test-Modus befinden, sichtbar durch einen roten Rahmen um den Editor. (Ist dies nicht der Fall wurde die Aufgabe schon einmal editiert und gespeichert, also befindet er sich in dem Modus in dem es weitergehen sollte). Seine Aufgabe ist es nun einen fehlschlagenden Test zu schreiben, oder dafür zu sorgen, dass der Code nicht compiliert. Wenn alles funktioniert sollte er mithilfe von Compile in den nächsten Modus kommen.

3. Jetzt befindet sich der Nutzer im, durch einen grünen Rand dargestellten, 'Simple-Code' Modus. Er muss nun in einer gewissen Zeit versuchen die Tests zu bestehen, sonst wird er automatisch zu den Tests zurückgeschickt. Der Nutzer kann aber auch manuell mit dem 'Step back' Button zurückkehren, falls es einen Fehler in den Tests gab. Schafft der Nutzer es den Test rechtzeitig zu lösen wird er mithilfe von 'Compile' in den Refactor-Mode gebracht.

4. Der Nutzer findet sich nun in dem, durch einen grauen Rand angezeigten, Refactor-Mode. Hier kann er seinen Code nach eigenem ermessen optimieren. Will jener zuerst testen ob ein Teil seines Codes funktioniert sollte er den 'Refactor' Button drücken, denn dieser zeigt einem nur mögliche Compilerfehler an ohne direkt zur Test-Phase zu wecheln. Ist der Nutzer fertig mit dem Refactor und es laufen alle Tests, kann er mithilfe des Compuile Buttons erneut in den Test-Modus wechseln.
5. Die Schritte 2-4 sollen solange wiederholt werden, bis das Feature vollständig implementiert wurde.
6. Hinweis: Akzeptanztests sind nicht wirklich implementiert deswegen ist der 'setAcceptanceTest' Button im Moment, nicht aktiviert.(Er kann aber über 'enableATTD' in der XML eingestellt werden)