# Benutzerhandbuch für die Test-Tutorial-Anwendung

Entwickelt von
Caro Kaschuba, Robert Kaluza, Michael Lorenz, Youssef Aissaoui, Tanja Roeder

Zuletzt aktualisiert am 14.07.2016

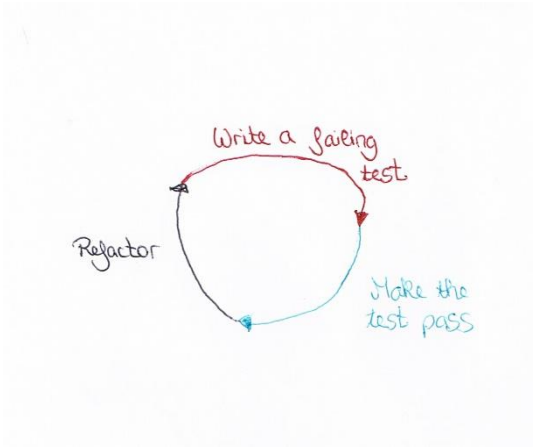
# Inhaltsangabe

- Kurzer Überblick
- Installationsbeschreibung
- User Gruppe
- Ziel und Aufbau der Anwendung
- Programmaufbau
- Beispiel
- Wichtige Informationen
- Zusammenfassung

# Kurzer Überblick

- Die Dozenten der Heinrich-Heine-Universität (kurz HHU) in Düsseldorf erhalten die Möglichkeit, alle von Studentinnen und Studenten (kurz: Studenten) zu bearbeitenden Programmieraufgaben als Aufgabenkatalog zu veröffentlichen
- Jeder Student erarbeitet die gestellten Programmieraufgaben eigenständig, die Reihenfolge der Aufgaben kann hierbei frei gewählt werden
- Selbst geschriebene Tests ermöglichen eine Überprüfung des geschriebenen Programms

- An bereits gearbeiteten Aufgaben können auch im Nachhinein Verbesserungen durchgeführt werden (Refactor)
- Ein Timer begrenzt allerdings das Schreiben von Tests und dem Programmieren, ist der Timer abgelaufen (nach 2-3 Minuten), werden Test und Code gelöscht



# Installationsbeschreibung

- Voraussetzungen für eine erfolgreiche Installation:
  - Betriebssysteme: Linux, Windows (Vista, 7, 8, 10), Mac OS X
  - 32- und 64- Bit System
  - Java Version 8+

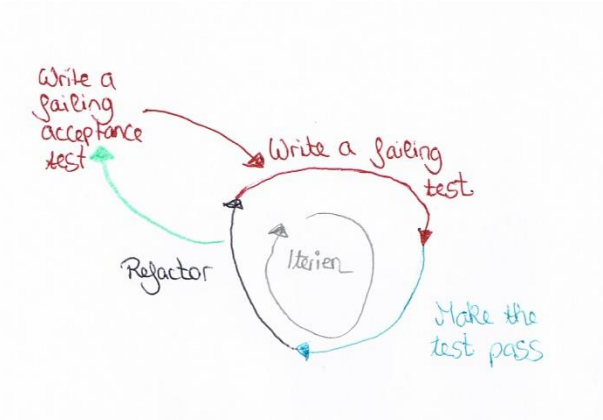
# User-Gruppe

- Anwendergruppe: das Programm ist speziell für die Erstsemester der HHU entwickelt worden
- Die Dozenten der HHU stellen die Aufgabenkataloge, die jederzeit verändert werden können (Hinweise zur xml Struktur, siehe Beispielaufgabe)

# Ziel und Aufbau der Anwendung

- Die Studenten müssen bereits vor der Implementierung ihres Programmes wissen, wie ihr Programm aufgebaut sein muss, also welche Ergebnisse es liefern sollte
- Anhand dieser Vorüberlegungen werden zunächst Tests geschrieben, die das spätere Programm erfüllen sollen
- Diese Tests werden unterteilt in einen großen (Akzeptanztest) und in mehrere kleine Tests
- Akzeptanztest: besteht aus mehreren kleinen Tests
- Tests: nacheinander wird ein Test hinzugefügt, sodass das Programm so lange verbessert werden kann, bis dieser Test funktioniert

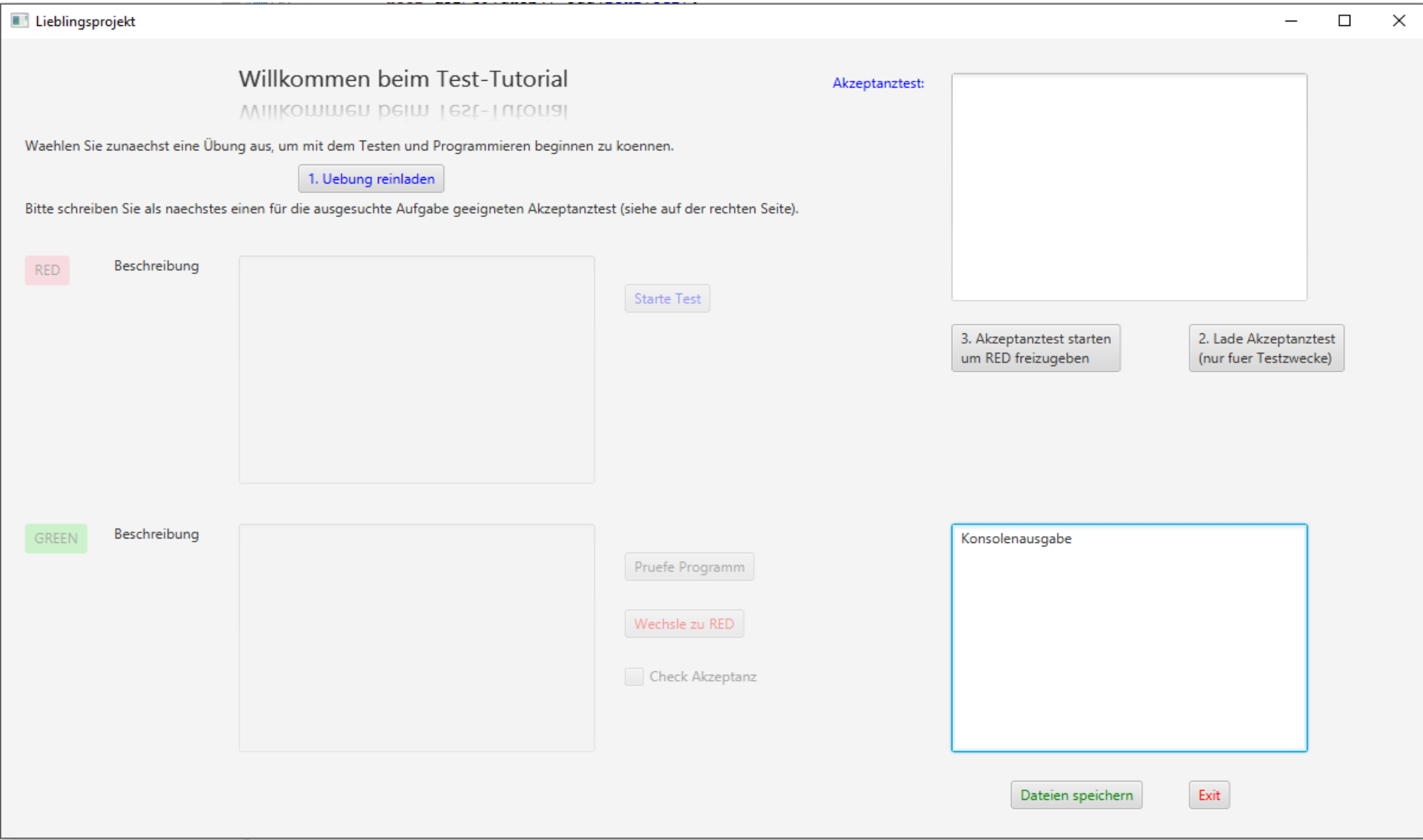
- Erst nach dem Hinzufügen des ersten Tests kann das Programm implementiert und getestet werden
- Beides, Test und Code, sind bei den Babysteps begrenzt zu bearbeiten, nach 2-3 Minuten löscht ein Timer alle bis dahin implementierten Tests und den Code
- Die kleinen Tests können immer weiter erweitert und an das Programm angeglichen werden (Refactor, vgl. Bild)



- Erst wenn auch der Akzeptanztest fehlerlos durchlaufen wird, ist das Programm korrekt
- Das Programm ist so innovativ, dass man weiß in welchem Feld was geändert werden muss

## Programmaufbau

# Startmaske



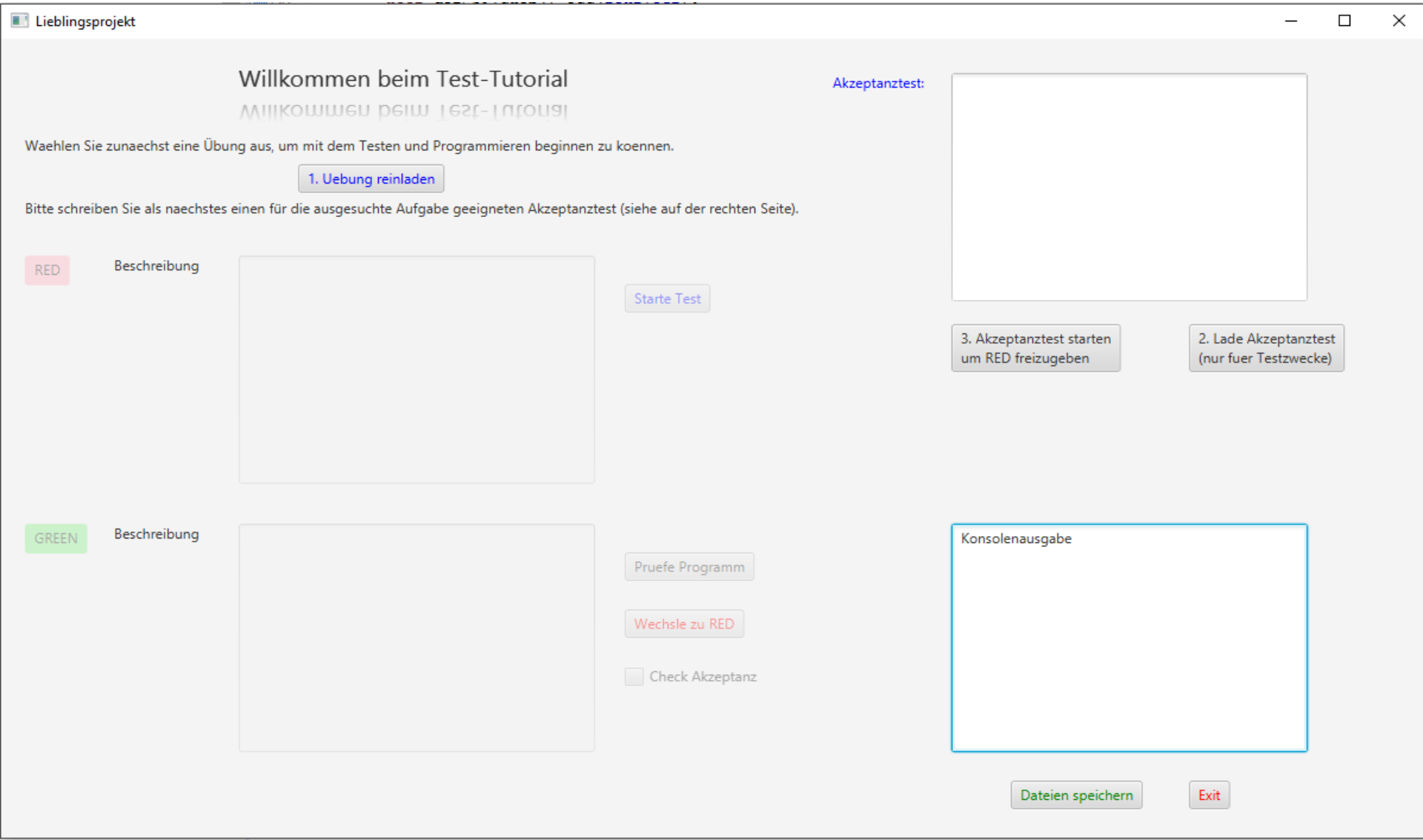
# Aktionen in der Startmaske

- **Button- „Uebung reinladen“** (links oben): hier wird vom Studenten die Aufgabe im sich öffnenden Fenster „Aufgabe“ ausgesucht, die als nächstes bearbeitet wird
- **Akzeptanztest** (rechts oben): beschreibt den gesamten Test für ein Programm, bestehend aus vielen kleinen Tests
- **Button „Akzeptanztest starten“** (rechts oben): das Programm (folgt später) wird mittels des Akzeptanztests durchlaufen und validiert; wenn ein Fehler in der Konsole (später) ausgegeben wird, geht es weiter mit Button-„RED“
- **Button- „RED“** (links Mitte): dieser schaltet das Text-Feld neben „Beschreibung“ frei, welches für kleinere Tests vorhanden ist; diese können im Laufe des Programmierens immer erweitert werden, im Gegensatz zum Akzeptanztest

- **Button- „Pruefe Programm“**: überprüft das Programm mittels der „kleinen Tests“, das Ergebnis wird in der Konsole ausgegeben
- **Button- „Starte Test“** (Mitte): durchläuft das Text-Feld mit dem Programm (folgt später) und testet es, wenn der Test erfolgreich war, erfolgt eine entsprechende Ausgabe in der Konsole, ansonsten kann der Button- „GREEN“ verwendet werden
- **Button- „GREEN“** (links unten): dieser Button schaltet das Text-Feld für das Programm frei
- **Programm**: der Student erhält hier eine Programm-Vorlage in die er sein Programm eingeben muss

- **Button- „Wechsle zu RED“**: es besteht die Möglichkeit nach einer Validierung des Programms zum „RED“ –Button zurückzukehren und bei den kleinen Tests wieder neue hinzuzufügen um diese auch im Programm zu berücksichtigen
- **Check-Button – „Akzeptanztest“**: das Programm wird mit dem umfangreichen Akzeptanztest durchlaufen, auch hier erfolgt die Ausgabe des Ergebnisses in der Konsole
- **Konsole** (rechts unten): Ausgabe sämtlicher Test-Ergebnisse mit entsprechenden Hinweisen bei Fehlern, was an der Ausgabe falsch ist
- **„Dateien speichern“ – Button**: zu jeder Zeit können die Dateien im selben Verzeichnis, in dem sich die Anwendung befindet, gespeichert werden

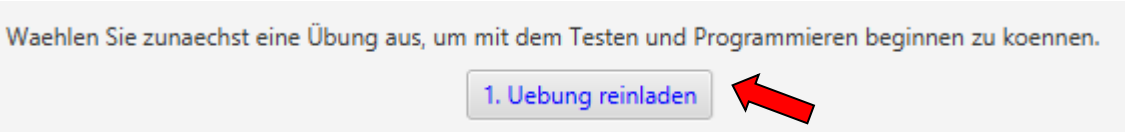
# Beispiel - Programm



1. Schritt:

# Übung reinladen

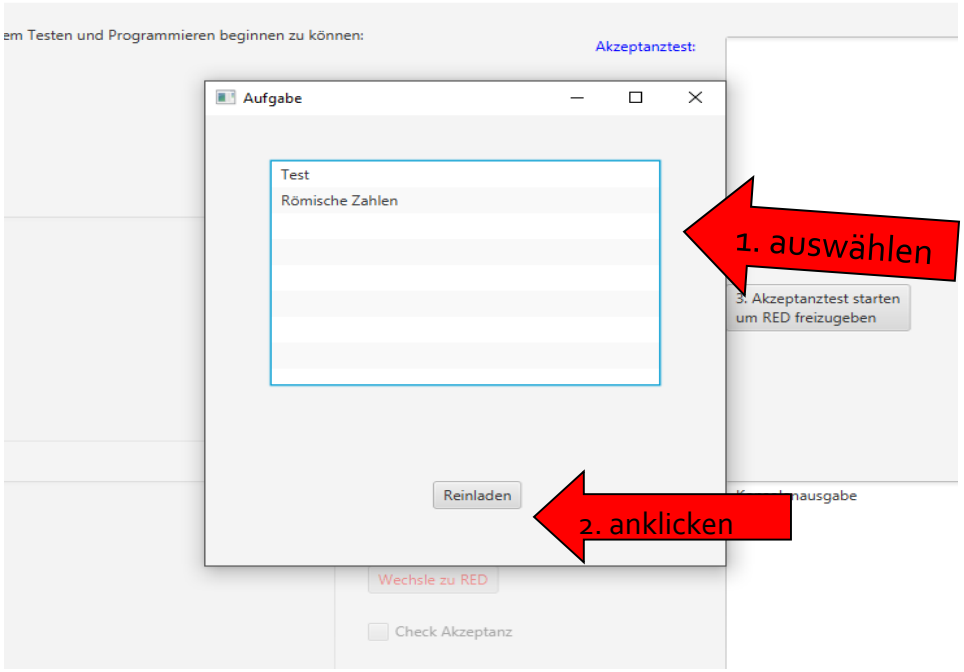
- rechts oben im Startfenster den Button „1. Uebung reinladen“ auswählen



## 2. Schritt:

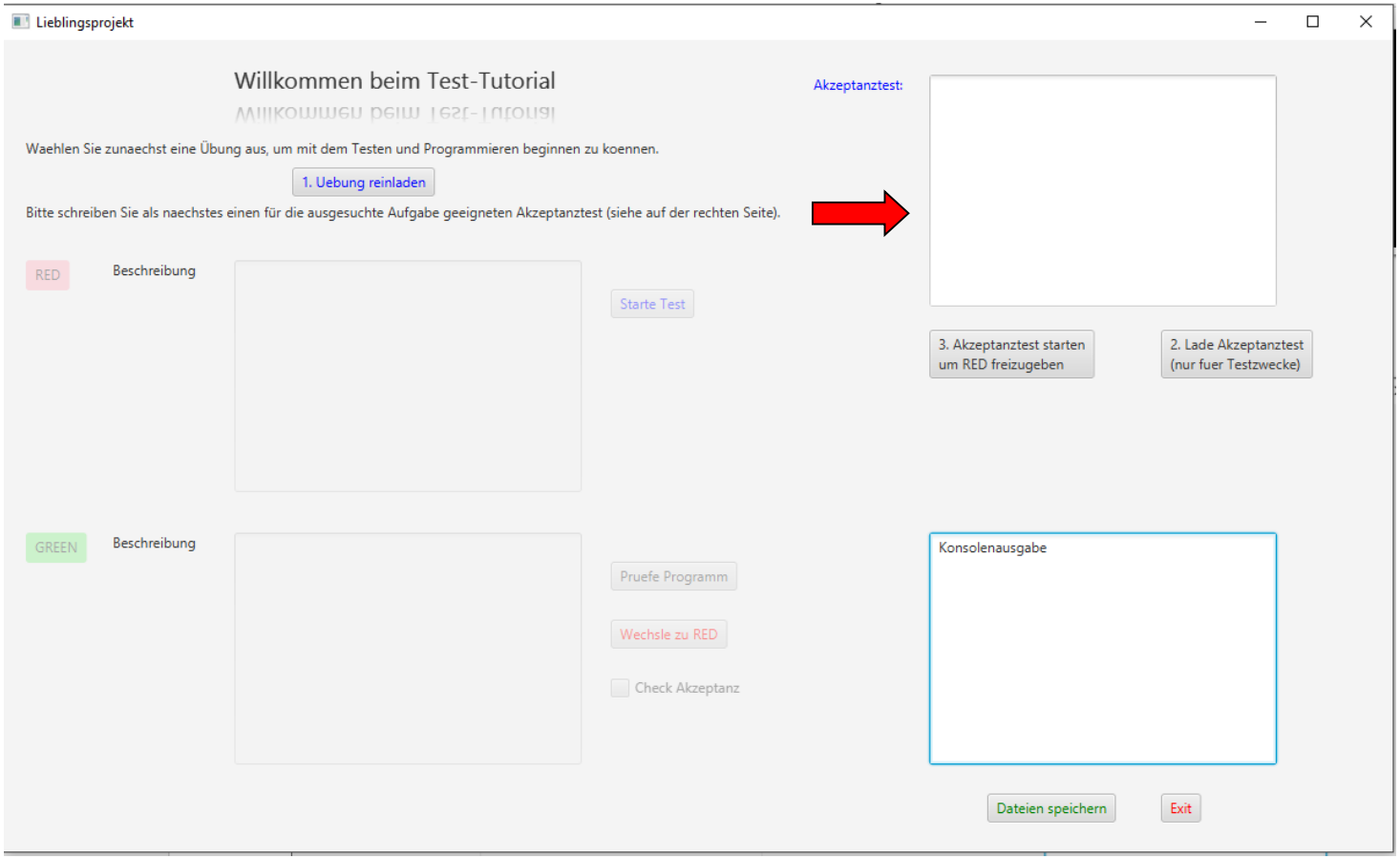
# Übung auswählen

- es öffnet sich das neue Fenster „Aufgabe“
- hier kann der Student die Aufgabe auswählen, die er als nächstes bearbeiten möchte
- anklicken der gewünschten Aufgabe
- danach auf „Reinladen“ klicken



# Startfenster: nach Wahl der Übung

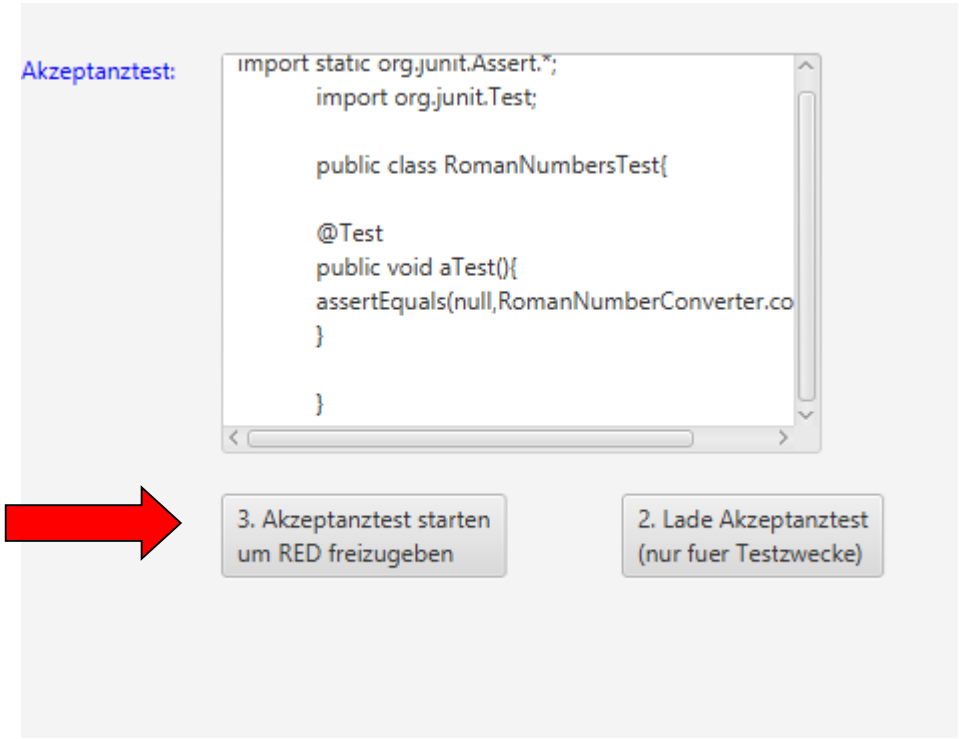
- das Text-Feld „Akzeptanztest“ wird editierbar



3. Schritt:

# Akzeptanztest

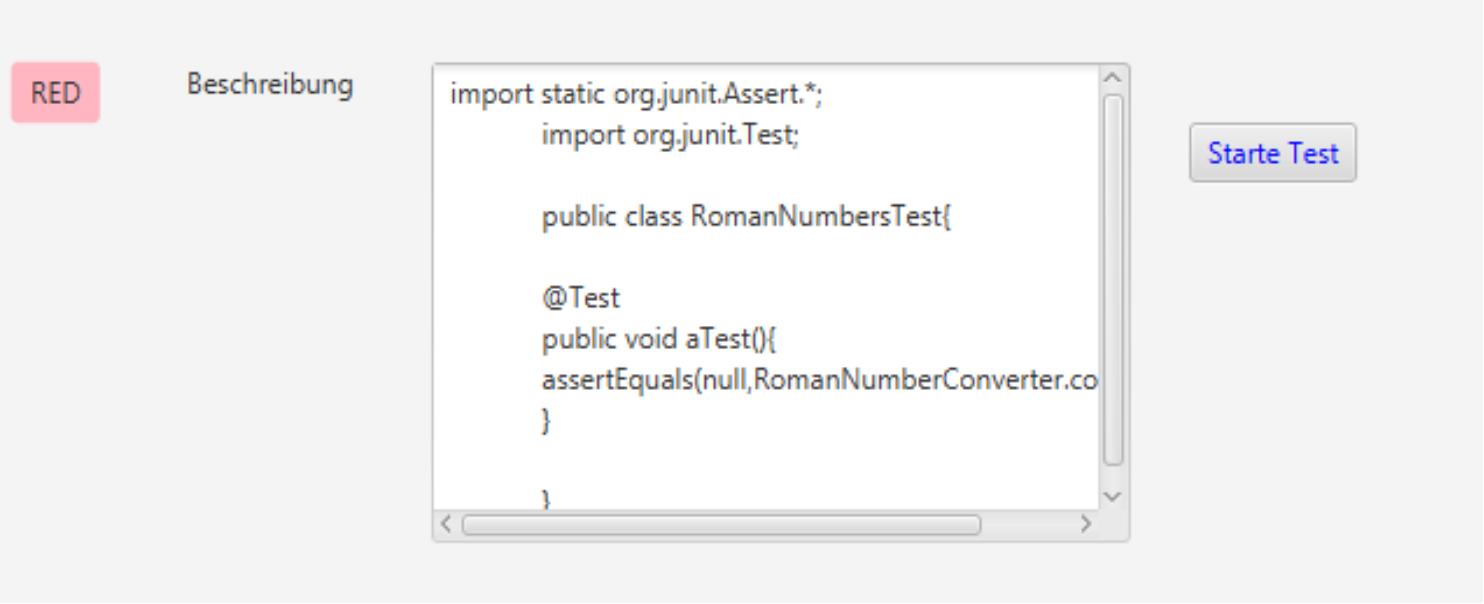
- im Text-Feld muss der Akzeptanztest implementiert werden
- mit dem Button „3. Akzeptanztest starten um RED freizugeben“
- hierdurch wird das Programm getestet, ob es den Akzeptanztest erfüllt
- wenn nicht, wird der „RED“-Button aktiviert
- wenn doch, gibt es in der Konsole den Ausgabe-Text „Akzeptanztest erfolgreich“



4. Schritt:

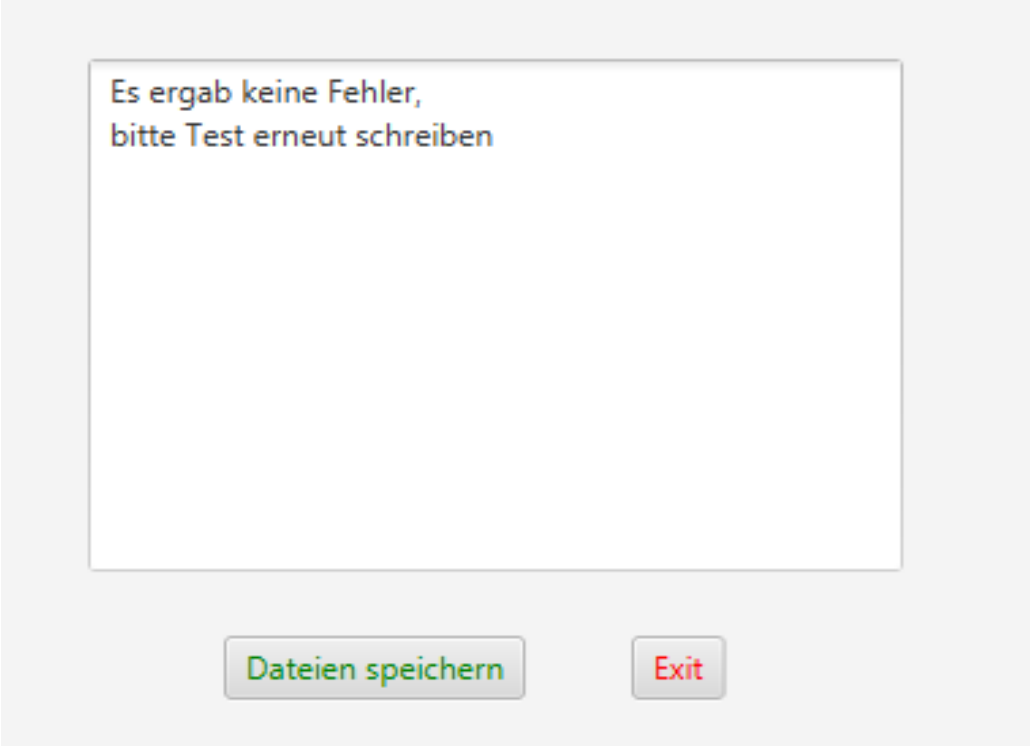
# RED-Button

- Der „RED"-Button aktiviert das Text-Feld für einen kleinen Test
- Dieser ermöglicht, dass das Programm erst einmal im „Kleinen" getestet wird
- Dementsprechend kann peu à peu ein neuer Test nach dem Anderen eingefügt werden, um das Programm langsam aufzubauen
- Mit „Starte Test" wird das Programm (in nächsten Text-Feld) getestet



# Test des Programms

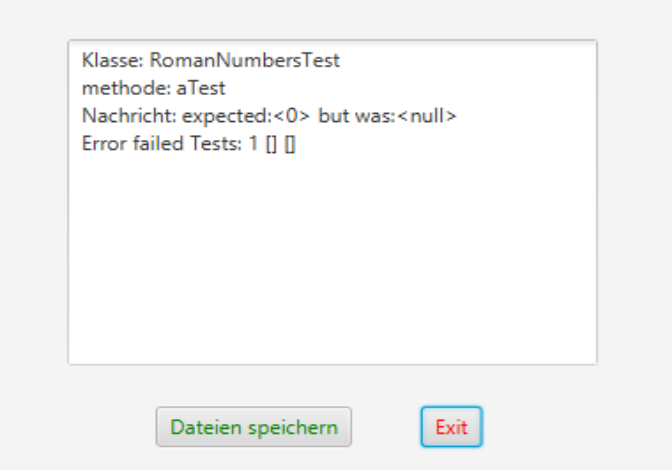
- Die Konsole zeigt an, ob das Programm erfolgreich getestet wurde
- Hier war der Test-Durchlauf erfolgreich und der alte Test kann um einen Neuen erweitert werden



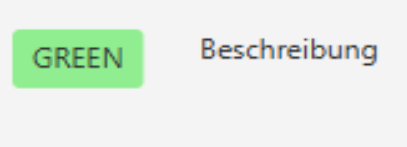
# Test des Programms



- Hier war der Test-Durchlauf nicht erfolgreich
- Die Ausgabe gibt an, was bemängelt wird



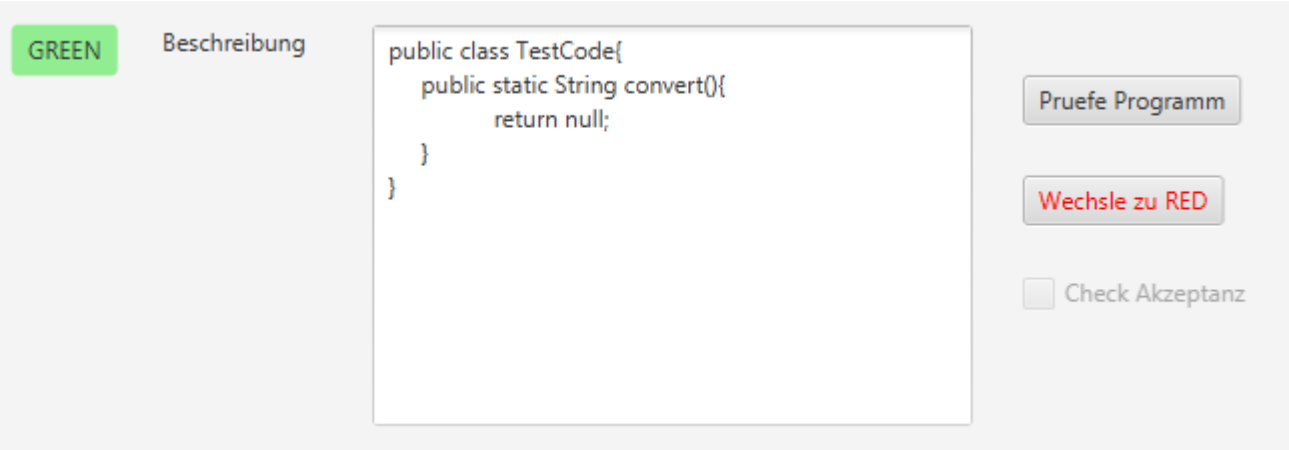

- Im nächsten Schritt wird deshalb der GREEN-Button editierbar



6. Schritt:

# Implementierung des Programms

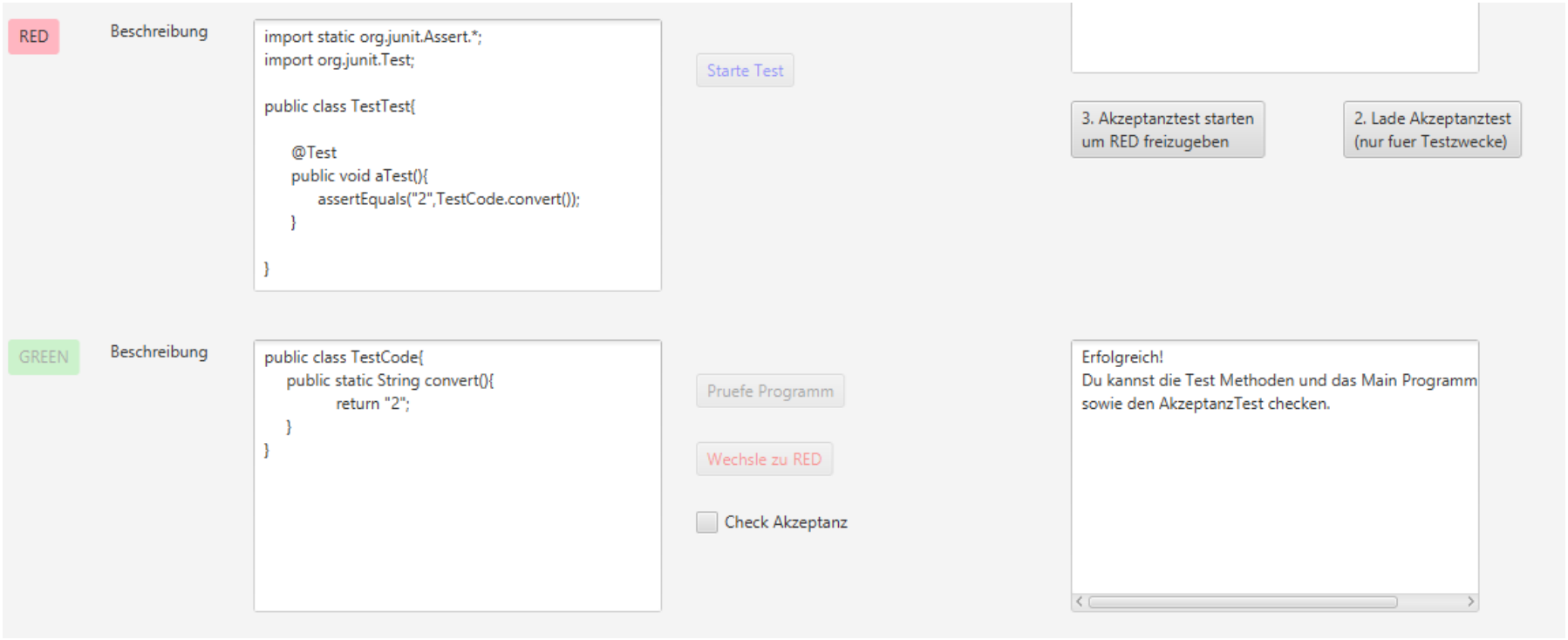
- In das Text-Feld nach dem GREEN-Button wird nun das Programm geschrieben
- Nachdem das Programm fertiggestellt wurde besteht die Möglichkeit es mit dem „Pruefe Programm"-Button zu testen
- Das Ergebnis wird auf der Konsole angezeigt



7. Schritt:

# Konsolenausgaben – erfolgreich

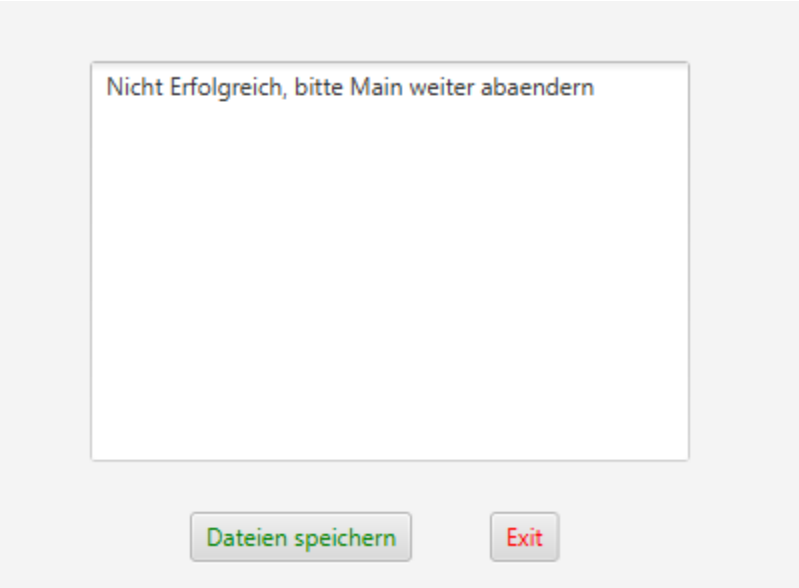
- Ausgabe nach dem erfolgreichen Test des Programms mittels des „Pruefe Programm"-Buttons
- Nun können weitere Tests in das Test-Text-Feld beim GREEN-Button hinzugefügt werden
- Zusätzlich kann der umfangreichere Akzeptanztest überprüft werden



8. Schritt:

# Konsolenausgaben – nicht erfolgreich

- Ausgabe nach dem erfolgreichen Test des Programms mittels des „Pruefe Programm“-Buttons
- Das Programm muss weiter angepasst werden



Wichtige Informationen:

# Dateien speichern

- Die Dateien können jederzeit mit dem „Dateien speichern"-Button in dem Verzeichnis, in dem auch die Test-Tutorial Anwendung liegt gespeichert werden


- Timer: Zwischen dem RED- und dem GREEN- Button befindet sich ein Timer (läuft im Hintergrund), der die Zeit zwischen dem Tests Schreiben und Implementieren des Programms auf 2-3 Minuten zur Übung begrenzt

# Zusammenfassung

Diese Anwendung sorgt für:

- Übersichtlichkeit
- Strukturiertes Denken
- Vorausschauendes Arbeiten der Studenten