Systembeschreibung: TDDT

- User Story
- Erweiterungen
- Aufbau
- CRC - Karten

User Story:

Der Nutzer startet die Anwendung TDDT und kann dann aus einem Katalog mit verschiedenen Übungen eine Übung auswählen. Die zugehörige Beschreibung erscheint auf der rechten Seite der Scene.
Hat der User eine Übung ausgewählt kann er durch den ‚select'-Button auf die nächste Scene gelangen. Auf dieser befindet sich auf der rechten Seite eine TextArea für das die rote Phase (Tests) und auf der linken Seite befindet sich eine TextArea für die Grüne Phase (Programm). In der Mitte befindet sich ein Textfeld für die Fehlerausgabe, sowie ein ‚zurück'- ‚weiter'- und ‚check'-Button. Abhängig von der Art der Übung kann ein Timer/ Uhr vorhanden sein.
Der Nutzer kann nun in die rote Phase einen Test schreiben, ist dieser fertig wird durch den Klick auf den ‚check'-Button geprüft, ob dieser Test fehlschlägt. Ist dies der Fall, erscheint der ‚weiter'-Button, durch den man das Textfeld der grünen Phase ändern kann. Hat man in der grünen Phase den zugehörigen Code geschrieben, damit der Test kompiliert, kann man dieses durch den ‚check'-Button prüfen. Kompiliert alles fehlerfrei, kann man in die schwarze Phase (Refactor) wechseln, in der man nun die Möglichkeit hat sowohl Test- als auch Programmcode zu ändern. Damit man einen erneuten Durchgang der Phasen durchführen kann, muss aber der Code und die Tests kompilieren. Der Nutzer hat jetzt die Möglichkeit soviel Durchgänge der Phasen rot, grün und schwarz durchzuführen wie er möchte. Wichtig sind dabei folgende Punkte:

- Es muss erst in der Roten Phase ein Test geschrieben werden, welcher nicht kompiliert, da die zu testende Methode/ Code noch nicht vorhanden ist
- Wurde dies erfolgreich durch den ‚check'-Button getestet, erscheint der ‚weiter'-Button und der Nutzer kann den Code der grünen Phase ändern. In die schwarze Phase gelangt der Nutzer nur, wenn der Test- und der Programmcode erfolgreich kompiliert
- In der schwarzen Phase kann sowohl Test-, als auch Programmcode geändert werden
- Durch den ‚zurück'-Button wird der Code der Phase, in der man sich befindet, gelöscht und man gelangt in die vorherige Phase zurück
- Der Nutzer kann nur den Code derjenigen Phase ändern, in der er sich zur Zeit befindet. Er kann den Code der anderen Phase aber sehen
- Ist in der Übung „Babysteps" aktiviert, so zählt ein Timer runter. In dieser Zeit muss dann der Test- / Programmcode geschrieben werden. Ist der Timer bei 0 angekommen und es wurde noch nicht auf den ‚weiter'-Button geklickt, so wird der Code gelöscht und man gelangt in die vorherige Phase. Auch hier zählt wieder ein Timer die Zeit runter
- Ist in der Übung „Tracking" aktiviert, so kann man jederzeit auf den ‚Chart anzeigen' Button klicken. Wird dieser betätigt, wird über ein Bar-Chart angezeigt, wieviel Zeit der Nutzer in den Phasen rot, grün und schwarz verbracht hat. Außerdem hat der Nutzer die Möglichkeit sich weitere Analyse Daten anzeigen zu lassen, indem er auf ‚Daten speichern' klickt

Erweiterungen:

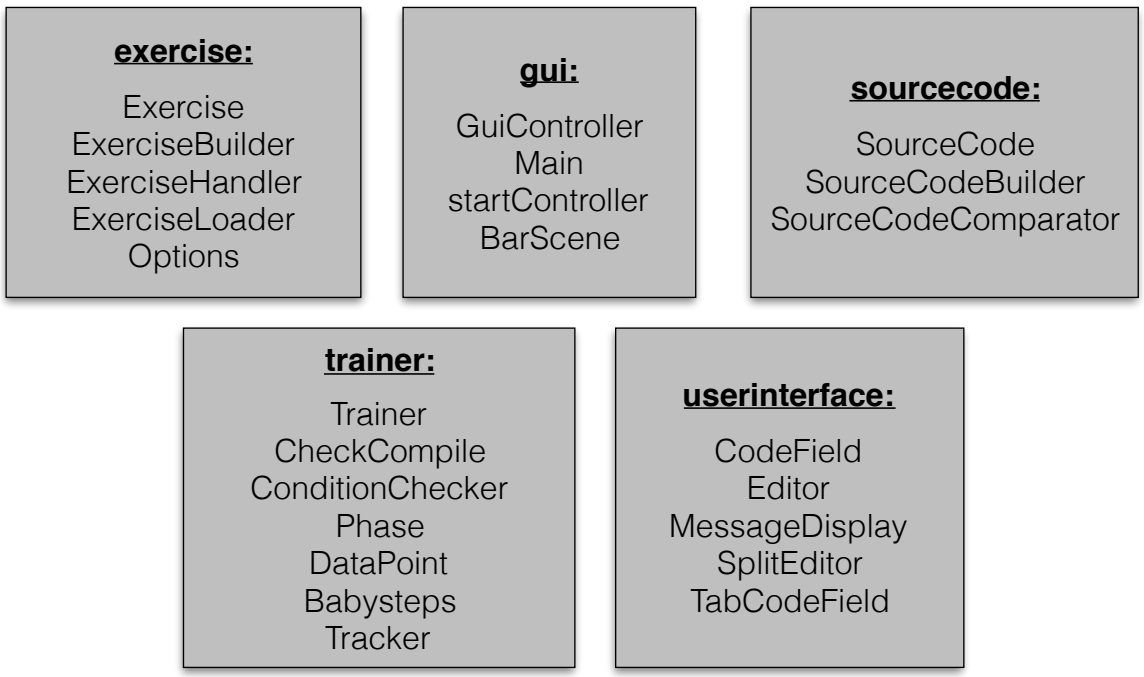
- *Babysteps*:
  - Ist die Funktion aktiviert, so zählt in der Phase, in der sich der Nutzer zur Zeit befindet, ein Timer runter
  - Kurz bevor der Timer abgelaufen ist, kommt ein Alert
  - Klickt man auf ‚check' und dann auf den ‚weiter' Button (Kompilieren war also erfolgreich) gibt es einen neuen Timer für die neue Phase
  - Ist der Timer abgelaufen und man hat es nicht geschafft auf ‚weiter' zu klicken, so wird der bereits geschriebene Code dieser Phase gelöscht. Wenn der Alert weggeklickt wurde, gelangt man automatisch in die vorherige Phase

- *Tracking*:
  - Ist die Funktion aktiviert, so gibt es einen Button ‚Chart anzeigen'
  - Ein Bar Chart erscheint mit der Anzeige, wieviel Zeit mit in den Phasen rot, grün und schwarz verbracht hat im Vergleich zur Gesamtzeit
  - klickt man auf den Button ‚Daten speichern', so werden die Analyse Daten in einer Textdatei gespeichert. Darunter befinden sich außer der Zeitanalyse auch weitere Informationen, die zuvor nicht graphisch dargestellt wurden
  - schließt man das Fenster, so wird das gesamte Programm geschlossen

Aufbau:

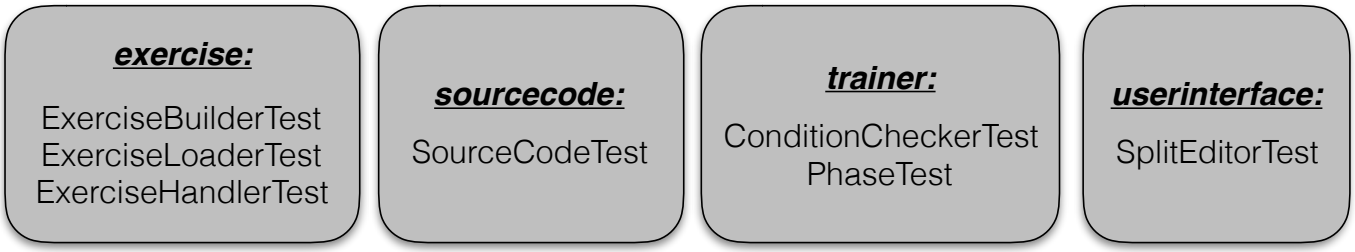
Ein grober Aufbau über das TDDT Programm liefert die folgende Darstellung. Genauen Überblick über die Funktionalität und Methoden der einzelnen Packages / Klassen verschaffen die CRC - Karten im nächsten Abschnitt.

src > main > java > de.hhu.propra16.tddt:



src > main > resources : fxml Dateien für die GUI

src > test > java > de.hhu.propra16.tddt:



CRC Karten:

src > java > de.hhu.propra16.tddt > exercise:

Klasse Exercise	**Interaktion/ Collaborators**
erzeugt Exercise Objekt bestehend aus Strings Name und Description, SourceCode Objekt und Details Objekt	SourceCode
getSources() -> SourceCode:	Options
getName() -> String:	ExerciseBuilder
getDescription() -> String:	
getOptions() -> Options:	

Klasse ExerciseBuilder	**Interaktion/ Collaborators**
erzeugt aus String ein ExerciseBuilder Objekt	Exercise
setDescription(String) / setTestName(String) / setClassName(String) / setClassCode(String) / setTestCode(String) / setBabySteps(boolean) / setTracking(boolean) *-> ExerciseBuilder*	SourceCodeBuilder
setTime(Duration) / setName(String)	
build() -> exercise:	

Klasse ExerciseHandler	**Interaktion/ Collaborators**
xml - Parser	Exercise, ExerciseBuilder

Klasse exerciseLoader	Interaktion/ Collaborators
erstellt aus einer xml-Datei Exercise Objekte	Exercise
load(InputSource) -> List<Exercise>: erstellt aus Katalog für jede Übung ein Exercise Objekt	ExerciseHandler

Klasse Options	Interaktion/ Collaborators
Objekt speichert Einstellungen für TDDT	Exercise
getTime() -> Duration	
getBabySteps() / getTracking() -> boolean	

src > java > de.hhu.propra16.tddt > sourcecode:

Klasse SourceCode	Interaktion/ Collaborators
erzeugt SourceCode Objekte	Trainer
hasCompileErrors() -> boolean	exercise
numberOfFailedTests() -> int	SourceCodeBuilder
getStringCode(String name) -> String : Bekommt einen Klassennamen übergeben und liefert den Inhalt als String zurück	
getNameTest() / getNameCode() -> List<String>: Gibt eine Liste zurück mit den Klassennamen der Test-Dateien / Code - Dateien	
getResult() -> String: gibt die Fehlermeldungen des SourceCode Objektes als einen String zurück	

Klasse SourceCodeBuilder	Interaktion/ Collaborators
build() -> SourceCode: erstellt SourceCode Objekt	SoureCode
add(CompilerUnit) -> SourceCodeBuilder: Fügt in List<CompilationUnit> ein Objekt hinzu	

src > java > de.hhu.propra16.tddt > trainer:

Klasse Trainier	**Interaktion/ Collaborators**
Regelt die Phasenwechselung, kümmert sich um BabySteps/ Tracking. Interaktion mit GuiController für die graphische Ausgabe von TDDT	SourceCode Exercise
checkPhaseStatus() / nextPhase() / previousPhase()	GuiController
	Tracker, Babysteps

Klasse ConditionChecker	**Interaktion/ Collaborators**
check(SourceCode, Phase) -> boolean: Prüft in der aktuellen Phase, ob Tests fehlschlagen, und ob in die nächste Phase gewechselt werden kann	CheckCompile SourceCode Trainer

Interface CheckCompile	**Interaktion/ Collaborators**
check(SourceCode, Phase) -> boolean	ConditionChecker

Klasse DataPoint	**Interaktion/ Collaborators**
enthält einige Getter / Setter Methoden für die Datenverarbeitung der Tracking Daten	Tracker

Klasse Babysteps	**Interaktion/ Collaborators**
Enthält Methoden für die Babysteps Zeit-Einstellungen	Trainer
timer(Trainer) / cancel()	

src > java > de.hhu.propra16.tddt > gui:

Klasse Main	**Interaktion/ Collaborators**
start(Stage):	Gui

Klasse StartController	**Interaktion/ Collaborators**
Graphische Darstellung der Exercise Objekte, lädt den Katalog	Main
	Exercise, ExerciseLoader

Klasse GuiController	Interaktion/ Collaborators
Verbindet Gui mit der Trainer Klasse	Trainer Gui

Klasse BarScene	Interaktion/ Collaborators
createBarScene(Tracker) -> Scene: erstellt Bar Diagramm für die Tracking - Analyse	

src > java > de.hhu.propra16.tddt > userinterface:

Klasse SplitEditor	Interaktion/ Collaborators
Hat zwei CodeFields für die Phasen und implementiert die Methoden von Editor	Editor

Klasse TabCodeField	Interaktion/ Collaborators
Implementiert die Methoden von CodeField	CodeField

Interface CodeField	Interaktion/ Collaborators
setEditable(boolean)	TabCodeField
showText(String name, text)	
getTextOf(String) -> String	
editNumber() -> IntegerProperty	

Interface Editor	Interaktion/ Collaborators
show(SourceCode,boolean, boolean)	SplitEditor
get() -> SourceCode	
changed() -> BooleanProperty	

Interface MessageDisplay	Interaktion/ Collaborators
show(String title, message)	