Bericht

Allgemeine Beschreibung

Das Programm TDDT (Test Driven Development Trainer) ist eine minimalistische Entwicklungsumgebung, die dem Nutzer helfen soll das sog. „Test Driven Development“ zu trainieren. Bei TDD geht es darum mit Hilfe von Tests den entwickelten Sourcecode zu prüfen und das möglichst kleinschrittig (Funktionsweise des Programms: Siehe Handbuch/Manual; Informationen zu TDD siehe http://www.frankwestphal.de/TestgetriebeneEntwicklung.html).
Es sind zwei Erweiterungen eingebaut: Babysteps und ATDD.
In **Babysteps** kann 1 - 5 Minuten, in Sekundenschritten, gewählt werden. Ist eine Länge gewählt, ist im Test Code und Source Code schreiben Zeitdruck angesagt! Wir haben darauf verzichtet den Akzeptanztest auch Zeitabhängig zu machen, da es für uns in diesem Schritt keinen Sinn ergab, da sich dieser Code besser übelegt sein sollte. Der Refaktormodus ist nach Aufgabenstellung ausgenommen.
In der Erweiterung **ATDD** (Acceptance Test Driven Development) werden zu den normalen Unit Tests im Testmodus noch weiterere Unit Tests implementiert. Man kann sich die Akzeptanztest als übergeordnete bzw. größere Tests vorstellen, die ein Feature eines Programms testen sollen und nicht kleinschrittig den Source Code wie die normalen Tests.

Interaktion von Klassen und Methoden

In der Klasse **TDDTMain** wird das Programm gestartet. Es wird die Ressource *layoutMain.fxml* genutzt, in der lediglich eine BorderPane erstellt wird. In diese werden in das Zentrum alle weiteren Ressourcen *layoutMenu.fxml*, *layoutTDDT.fxml* und *layoutATDD.fxml* hineingeladen. Es wird eine einzelne Stage für das gesamte Programm genutzt.

In der Klasse **LayoutMenuController** wird das Startmenü erstellt. Zu Anfang wird die Methode „*initialize()*“ gestartet und es werden automatisch alle .txt Dateien aus dem Ordner „Aufgaben“ in eine ListView geladen, dazugehörig eine ObserableList. Eine (oder mehrere auf einmal) externe .txt Datei/en zu laden ist ebenfalls möglich. In dem Menü kann man die jeweiligen Erweiterungen an- oder abschalten und die Zeit für Babysteps einstellen. Ein besonderes Augenmerk bitten wir auf den Button „Hübsch“ zu legen, der die GUI „verschönert“ (was natürlich Geschmacksache ist!).

In den Klasse **LayoutTDDTController** und **LayoutATDDController** (zweite erbt von ersterer) startet man in die GUI für das eigentliche TDD. Zu Anfang wird in beiden Klassen die Methode „*initialize()*“ gestartet. Es werden public Methoden für die Handles für die jeweiligen Button erstellt und private Methoden, die Anweisungen auslagern und speziell die „*HandleRunButton()*“ und „*HandleRefactor()*“ übersichtlicher machen sollen.
In den drei setter-Methoden werden entsprechend die Labels, etc passend zu der jeweiligen Phase gesetzt (Beispielsweise in Phase Rot den TestCode editierbar machen).

In der Klasse **Phases** mit dem Interface **InterfaceTDDT** ist die interne Logik mit Herrn Bendispostos Bibliothek verknüpft und wird genutzt um die jeweiligen Compiler Errors zu bekommen und zu nutzen und die Phasen „red“,“green“,“refactor“ und „akzeptanz“ zu setzen.

In der Klasse **Babysteps** wird für die Erweiterung Babysteps der Timer mit Hilfe einer Timeline initialisiert. Dort wird der Timer gestartet, gestoppt und geresettet werden.

Klassen und Resources

Paket	Klasse	Resources
babysteps	Babysteps.java	
phases	IntefaceTDDT.java Phases,java	
tddtcycle	TDDCycle.java	
tddtlayout	LayoutMenuController.java LayoutTDDTController.java LayoutATDDController.java	layoutMenu.fxml layoutTDDT.fxml layoutATDD.fxml
tddtMain	TDDTMain	layoutMain.fxml
katalog	Katalog.java	
		tddt.css

Klassen und Methoden

Klasse	Methode
Babysteps	- public Babysteps (int max_time, Function callback) - public void reset() - public void start() - public void stop()
Phases	- public Phases(String phase) - public String getPhase() - public void setPhase (String phase)
TDD	- public TDDCycle(String phase) - public void compile (String code, String test) - public void compile (String akzeptanz, String code, String test) - public Collection<CompileError> getCompileErrorsTest() - public Collection<CompileError> getCompileErrorsAkzeptanz() - public Collection<CompileError> getCompileErrorsCode() - public boolean hasCompileErrors()

	- public boolean hasFailingTest()
LayoutMenuController	- public void inititalize() - public void handleBabystep() - public void handleExitButton() - public void handleStartMenuButton() - public void handleNewExercise() - publi void rainbow() - private StringBuilder readTxt(String file) - private void viewExercise() - private void aufgabenErstellen() - static int getTimer() - static boolean getBabysteps() - static String getExerciseTest() - static void setHasAtdd(boolean b) - static void setHasBabysteps(boolean b) - public static void setHasRainbow(boolean Rainbow)
LayoutTDDTController	- public void initialize() - private Object resetCode(Object o) - void setPhaseRed() - void setPhaseGreen() - void setPhaseRefactor() - public void handleRunButton() - private boolean hasNewTest() - public void handleBackToTestsButton() - public void handleBackButton() - public void handleRefactor() - protected void tddRefactoring()
LayoutATDDController	- public void initialize() - public void handleRunButton() - public void handleRefactor() - public void handleAcceptance() - private void atddRefactoring() - private void isNewAcceptancePhase() - private void setPhaseAcceptance() - private void accomplishAcceptanceTest() - private void chooseLastPhase() - private void setToNormal()
TDDTMain	- public static void main(String[] args) - public void start(Stage primaryStage)