Systembeschreibung TestDrivenDevelopmentTrainer

Inhaltsverzeichnis

1 Offene Fragen

2 Systembeschreibung

2.1 Herangehensweise an das Projekt

2.2 Systemstruktur und Klasseninteraktion

1 Offene Fragen

Frage: Wie geht der Phasenwechsel von RED zu GREEN bei Compilefehlern von statten?
Antwort: Der Wechsel ist nur möglich wenn es nur einen Error im Testcode gibt und dieser Error muss der cannot find symbol: method[…] Error sein. Also darf nur die zu testende Methode nicht vorhanden sein, wenn es nicht kompiliert. Um zu überprüfen, ob es sich um diesen Error handelt, wird die Compilemessage auf die Wörter „find" , „symbol" , „symbol:" und „method" hin überprüft.

Frage: Wie funktioniert der Aufruf von Exercises im Zusammenhang mit Projekten?
Antwort: Wenn eine Exercise ein Mal aufgerufen wurde und es dementsprechend ein Projekt mit dem gleichen Namen im Projektordner gibt, dann kann man die Exercise nicht erneut öffnen. Stattdessen öffnet sich ein Alert mit einer entsprechenden Message. Man kann dann nur das angefangene Projekt öffnen.

Frage: Wie werden Projekte gespeichert?
Antwort: Projekte müssen bei jedem schließen gespeichert werden. Sie werden dann im Homeverzeichnis im Ordner TestDrivenDevelopmentTrainer gespeichert. In diesem Ordner befindet sich ein Ordner mit dem Namen des Projektes, der die die Java Dateien und Logs für das Tracking beinhaltet.

Frage: Wann werden Babysteps aktiviert?
Antwort: Hat man Babysteps in einer Phase aktiviert, so laufen diese ab dem Wechsel zur nächsten Phase. Dort läuft dann der Timer runter. Allerdings ist zu beachten, dass es in Refactor keine babysteps gibt, da es nicht dem Sinn, sowohl des Refactoring, als auch des Babystepping erfüllen würde.

2 Systembeschreibung

Zunächst wird die Herangehensweise an das Projekt erläutert, also Überlegungen, wie man das Programm aufteilen sollte und Andere. Daraus ergibt sich schon die grobe Systemstruktur, die auch die Klasseninteraktion und die Funktionsweise des Projektes enthält.

2.1 Herangehensweise an das Projekt

Wir haben uns zunächst einzeln Gedanken gemacht über den groben Aufbau des TDDT, also welche Klassen benötigt werden und wie diese zusammenarbeiten sollten. Diese Gedanken haben wir dann beim zweiten Treffen zusammengetragen und damit eine vorläufige Struktur festgelegt. Diese hat sich dann aufgrund verschiedener Faktoren geändert unter Anderem hatte die Konversion von und für Gradle die Packages angepasst, wodurch sich dann die endgültige Struktur ergeben hat.

2.2 Systemstruktur und Klasseninteraktion

Es gibt zunächst im src-Ordner die Packages resources und test, wobei resources die fxml, css, png und auch die Manual html Dateien, die vom Programm neben den Java Dateien zur Ausführung benötigt werden beinhaltet und test die entsprechende Tests für die Klassen aus dem main.java.tddt Package.
Im folgenden werden die wichtigsten Klassen, die alle zusammenarbeiten, sowie deren Funktionsweisen und Zusammenhänge kurz erläutert.

Controller:
Der Kontroller steuert die Anwendung und ruft entsprechend die Methoden auf, je nachdem, welcher Knopf gedrückt wurde. Hierbei ruft er Methoden aus allen anderen wichtigsten Klassen auf, wie etwa Coordinator, ProjectIO oder LogList, wobei er direkt die Controllerklassen im Dialogsordner benötigt.

Coordinator:
Im Coordinator ist die gesamte Logik des TDDT geschrieben. Hier

werden Phasenwechsel koordiniert, kompiliert und die Zeit des Timers genutzt, sowie alle anderen Funktionen mit Babysteps and Andere.

Log und LogList:
Diese beiden Klassen sind für die Speicherung aller relevanten Daten zuständig. In einem Log wird der Inhalt der Klasse und des Tests, die Uhrzeit, die Phasenzeit und die Compilemessage gespeichert. Mit der LogList werden Logs gesammelt und die Methoden zum Laden von Logs aus xml-Dateien und zum Speichern der Logs in xmls zur Verfügnung gestellt. Abgesehen davon werden noch Methoden zum Löschen und zurückgehen für die TDDT Logik bereitgestellt.

Exercise:
Diese Klasse dient zum Erstellen und Laden von Exercises aus und in xml-Dateien.

ProjectIO:
Diese Klasse ist für das Laden und speichern von angefangenen Projekten zuständig. Sie lädt also alle benötigten Dateien aus dem TDDT Ordner mit dem Namen des Projektes, das man mit dem Filechoose ausgewählt hat.

Timer:
Der Timer ist sowohl für Tracking, als auch für die Babysteps nötig. Hier wird die Zeit gemessen, die unter Anderem im Trackinggraph zu sehen ist und die ablaufende Zeit, falls Babysteps aktiviert sind, wobei, wenn die Zeit abläuft die entsprechende Methode im Coordinator aufgerufen wird.