TDDT-Systembeschreibung

Unsere Klassen :

Für TDDT haben wir insgesamt **12 Klassen** implementiert.
Im Folgenden werden die Klassen kurz beschrieben und daraufhin die wesentlichen Funktionsweisen und Interaktionen erklärt.
Wir haben uns bei der Implementierung von TDDT darauf geeinigt, Babysteps und Tracking zu implementieren.

- ***TDDT*** – unsere "Main", von hier aus wird alles "in Gang" gesetzt.
- ***UserCode*** – verwaltet Nutzer-Eingaben, bzw. den vom Nutzer eingetippten Code.
- ***ActionUnit*** – kompiliert und testet UserCode, prüft auf Kompilierfehler und Testfehler.
- ***Controller*** – sorgt für den Eintritt der Phasen (RED,GREEN,REFACTOR) und stellt diese visuell dar.
- ***StartController*** – zeigt und verwaltet den Übungskatalog.
- ***Uebung*** – analysiert in Textdateien geschrieben Übungsdateien, speichert die notwendigen Informationen und gibt sie an StartController weiter.
- ***GreenValidator*** – überprüft auf einen gültigen Phasenwechsel von RED nach GREEN.
- ***PhaseSetter*** – hilft Controller dabei, die Phasenwechsel anzuzeigen.
- ***WarningUnit*** – zeigt die verschiedensten Fehler an, die beim Laufen von TDDT eintreten könnten, ist sozusagen eine Hilfsklasse für den Nutzer.
- ***TrackingUnit*** – speichert aktuelle Informationen über das Geschehen beim Laufen von TDDT, erzeugt eine visuelle Darstellung der Daten.
- ***TrackingTable*** – hilft TrackingUnit dabei, die Tracking-Daten zu speichern.
- ***BabyStep*** – verwaltet eine Zeitlimitierte Anwendung von TDDT.

Fragen, die wir uns gestellt haben :

Wie genau soll das Programm ablaufen ? Sind die Phasenwechsel vollautomatisiert ?

Wir haben uns dazu entschieden, die Phasenwechsel durch Buttons zu realisieren.
Der Nutzer wird darum gebeten, seinen Code einzugeben. Ist er der Ansicht, dass er für die jetzige Phase fertig ist, kann er durch Klicken der Buttons seine nächst gewollte Phase aussuchen, natürlich nur, wenn auch die Bedingungen gegeben sind.

Soll beim Wechsel von RED nach GREEN nur der Test-Code nicht kompilierbar sein oder der Test- und der Green-Code ?

Wir haben uns dazu entschieden, dass beide Codes kompiliert und getestet werden.
Sollte sich hierbei ein Kompilierfehler ergeben, wird nach GREEN gewechselt.

Was passiert, wenn der Nutzer beim Wechsel nach REFACTOR kompilierfehler hat ?
Soll dann bei einem möglichen Wechsel nach RED der Code in GREEN gelöscht werden ?

Der Nutzerfreundlichkeit halber haben wir uns dazu entschieden, den Code nicht zu löschen.

Was ist, wenn der Nutzer Übungen falsch editiert ?

Mithilfe von WarningUnit werden entsprechende Fehlerbehandlungen durchgeführt.

Wie sollen die Übungsdateien aussehen und ist *es erforderlich die Erweiterungen (Babysteps & Tracking) im Übungskatalog als Konfigurationsoptionen einzubauen ?*

Eine Formatvorlage haben wir Angehängt.
Desweiteren haben wir uns dazu entschieden, im Falle von Babysteps, den Nutzer nach Auswahl einer Übung zu fragen, ob er denn Babysteps aktivieren möchte.
Im Falle von Tracking steht dem Nutzer durch freiwillige Betätigung des entsprechenden Buttons völlig frei, ob er Trackingdaten einsehen möchte.

Was ist wenn ...?

Es gibt noch mehrere Situationen, wo Fehler auftreten können.
Für diese Fälle haben wir Meldungen eingebaut, welche dem Nutzer mitteilen was nicht stimmt.

Funktionsweise & Interaktionen

Die Klasse TDDT:
Hier werden die entsprechenden Instruktionen ausgeführt, um die Anwendung zu starten. Es wird die benötigte FXML geladen, sowie StartController und Controller miteinander verknüpft.
Es handelt sich somit bei TDDT um eine Art "Schaltzentrale".

Die Klasse UserCode:
Eine Klasse, die den vom User eingegebenen Code speichert und verwaltet.
Hierbei sind gleich die zwei unterschiedlichen Implementierungen des Nutzers gespeichert, nämlich die der Tests bei RED und die des zu testenden Codes bei GREEN.

Die Klasse ActionUnit:
Kompiliert und testet UserCode mithilfe der von Jens mitgegebenen Bibliotheken.
Gibt Auskunft darüber, ob ein Code kompiliert werden konnte oder nicht und ob die Tests erfolgreich waren.

Die Klasse Controller:
Im Grunde das "Herzstück" von unserer Anwendung.
Stellt im Prinzip die GUI dar und verwaltet die Phasenübergänge mithilfe von UserCode und ActionUnit.
Weiterhin werden hier mögliche Fehler mithilfe von WarningUnit angezeigt.

Die Klasse StartController:
Ist verantwortlich für die Darstellung des Übungskatalogs.
Bekommt Daten von Uebungen und leitet diese an Controller weiter.

Die Klasse Uebung:
Analysiert den beigefügten Übungskatalog und speichert die entsprechenden Informationen der Übungsaufgaben und leitet diese an StartController weiter.

Die Klasse GreenValidator:

Ist speziell für den Phasenwechsel von RED nach GREEN verantwortlich.
Sorgt dafür, dass die geforderten Bedingungen erfüllt sind, um einen Phasenwechsel durchzuführen. Arbeitet insbesondere mit Controller zusammen.

Die Klasse PhaseSetter:
Ist eine kleine Hilfe für Controller. Ist verantworlich für eine Visualisierung der Übergänge zwischen den einzelnen Phasen z.B durch Einfärben von Buttons.
Arbeitet natürlich mit Controller zusammen.

Die Klasse WarningUnit:
Sorgt für die Anzeige von Hinweisen, Fehlern und Kompilier- bzw. Testfehlern.
Ist eine Hife für den Nutzer, da er so bestimmte Fehler einsehen kann.
Ist im Prinzip über die ganze Anwendung "verteilt", überall da, wo es Probleme geben könnte.

Die Klasse TrackingUnit:
Speichert alle Geschehenisse beim Laufen von TrackingUnit in einer TrackingTable.
Misst außerdem die Zeit in den Phasen und stellt diese in einem Tortendiagramm dar.

Die Klasse TrackingTable:
Speichert alle Aktionen (wie z.B Code konnte nicht kompiliert werden) , sowie die zugehörige Zeit. Stellt all diese Informationen der TrackingUnit bereit.

Die Klasse BabySteps:
Stellt im Prinzip einen Timer dar. Sollte die konfigurierte Zeit abgelaufen sein, wird mithilfe von Controller zur letzten Phase gewechselt.

Die wichtigsten Grundsteine zur korrekten Ausführung unseres Programms finden sich in Controller. Wie oben beschrieben, wird hier mithilfe der oben beschriebenen Klassen überprüft, ob und wann ein Phasenwechsel gültig ist oder nicht.