Systembeschreibung

Programmaufbau

Die Graphische Oberfläche wird über die MainWindow.fxml dargestellt, die Logik geht über die MainWindowController Klasse. Von hier aus werden alle Objekte und Klassen angesprochen.

Der Katalog besteht zur Zeit aus 4 Übungsblättern mit mehreren Übungen, die alle jeweils aus einem Code und einem Test-Teil bestehen. Die Übungen sind im XML Format. Der Katalog kann beliebig erweitert werden.

Kompiliert wird durch die virtual-kata-lib. Der Compiler output wird dabei über eine TextArea am unteren Rand des Programms dargestellt. Falls es zu Kompilierungsfehlern kommt, werden diese durch die ErrorCounter Klasse gezählt und können durch ein Klick auf den “Tracking Data” Button angezeigt werden. Zusätzlich wird die Zeit gespeichert, die der User in den einzelnen Phasen zugebracht hat und graphisch dargestellt (Absolute Werte und relative durch Klick auf das Diagramm). Über den Filehandler können Klassen als normale .java Dateienmit in die Kompilierung eingebunden werden. Beispielhaft hier mit StdRandom und StdOut.

Eine XML besteht wie vorgegeben aus mehreren Exercises. Diese Exercises beinhalten den Namen der Exercise, Beschreibung, vorgegebenen Klassen und Tests und Config, die angibt, ob Timetracking und/oder Babysteps(+ Zeitlimit) aktiviert sind für die jeweilige Übung.

Der BabystepTimer (falls aktiviert) zählt in den Phasen RED und GREEN von der übergebenen Zeit runter. Wenn 0 erreicht wird, wird eine Meldungs ausgegeben, der Code auf den ursprünglichen Zustand zurückgesetzt und der Timer neugestartet.

Die Charts werden mit Daten aus dem TimeKeeper und ErrorCounter berechnet. Für die Darstellung werden die javafx internen Klassen PieChart und BarChart genutzt.

Das Textfeld ist mit Hilfe von Tomas Mikulas RichTextFX (https://github.com/TomasMikula/RichTextFX)
Implementierung eingebaut. Es wird Syntax Highlighting unterstützt (JavaKeywordsAsync und java-keywords.css). (Implementiert mithilfe der Beispiele im RichTextFX Demo repository)

Die Projekt Struktur setzt sich aus den Unterordnern “objects”, “panes” und “util” zusammen. In “objects” befinden sich alle Objekte, in “panes” die verwendeten panes und in “util” Helferklassen für den MainWindowController.

User Story

User Story 1 (Aufgabe bearbeiten ohne Babysteps)

Der User muss das Programm TDDT bereits geöffnet haben.

1. Der User öffnet per File/Open exercise das zu bearbeitende Blatt.
2. Mithilfe des nun geöffneten Menüs lässt sich die Aufgabe auswählen, welche zu bearbeiten ist.
3. Zunächst wird ein Test geschrieben, da man sich zunächst in RED befindet. Sobald man einen fehlschlagenden Test hat, kann man zu GREEN wechseln.
4. In GREEN kann der User nun seinen Code bearbeiten und notfalls zurück zu RED
    a. Falls der User zurück zu RED wechselt wird der CODE aus GREEN gelöscht.
    b. Ansonsten kann der User, falls es keine fehlschlagenden Tests gibt und das Programm compiliert weiter zu **Refractor.**
5. In **Refractor** kann der User seinen Code optimieren und dann anschließend , falls immernoch alle bisherigen Tests fehlerfrei laufen und der Code compliert weiter zu RED um einen neuen Test zu schreiben.

User Story 2 (Aufgabe bearbeiten mit Babysteps)

Der User muss das Programm TDDT bereits geöffnet haben.

1. Der User öffnet per File/Open exercise das zu bearbeitende Blatt.
2. Mithilfe des nun geöffneten Menüs lässt sich die Aufgabe auswählen, welche zu bearbeiten ist.
3. Zunächst wird ein Test geschrieben, da man sich zunächst in RED befindet, nun fängt der Babysteptimer an runter zu zählen. (Die Zeit wird in der Aufgabe vorgegeben)
4. In GREEN kann der User nun seinen Code bearbeiten und notfalls zurück zu RED. Die Bearbeitung muss nun in der vorgegeben Zeit abgeschlossen werden, ansonsten wird der Code gelöscht und wechselt zurück zur letzten Phase.
    a. Falls der User zurück zu RED wechselt wird der CODE aus GREEN gelöscht.
    b. Ansonsten kann der User, falls es keine fehlschlagenden Tests gibt und das Programm compiliert weiter zu **Refractor.**
5. In **Refractor** kann der User seinen Code optimieren und dann anschließend , falls immernoch alle bisherigen Tests fehlerfrei laufen und der Code compliert weiter zu RED um einen neuen Test zu schreiben.
   Für das die **Refractor-Phase** gibt es keinen Zeitlimit.