Klassenbeschreibung

main

Main	java	Einstiegsklasse des Programms, erweitert die Klasse "Application" von JavaFX und zeigt ein Fenster an, in das die Hauptansicht geladen wird. Siehe die Klassen "FxmlLoader", "AttdController", "AttdModel" und die Datei "AttdView.fxml".

core

Catalog	java	Repräsentiert einen Katalog von Aufgaben, der einen Namen besitzt und eine Liste von Aufgaben (in einer Liste von Instanzen der Klasse "Exercise") speichert.
Exercise	java	Repräsentiert eine Aufgabe, die einen Namen und eine Beschreibung, die Konfiguration einer Aufgabe (mit der Klasse Configurations"), die vordefinierten Codes einer Aufgabe (mit der Klasse "Codes"), sowie die Versionen (in einer Liste von Instanzen der Klasse "Version"), die der Benutzer angelegt hat, speichert.
Version	java	Repräsentiert eine Version einer Aufgabe des Benutzers. Eine Version besitzt einen Titel und die Codes die der Benutzer eingegeben hat (gespeichert mit der Klasse "Codes").
Configurations	java	Wird von der Klasse "Exercise" verwendet und speichert die Konfiguration einer Aufgabe, d.h. ob Akzeptanz-Tests und Baby-Steps aktiviert sind und ggf. wie viel Zeit der Benutzer zur Bearbeitung mit Baby-Steps hat.
Codes	java	Wird von den Klassen "Exercise" und "Version" verwendet, um die Codes einer Aufgabe zu speichern: den Testcode, den Klassencode (beides mit der Klasse "Code") und, wenn nötig, den Akzeptanzcode (mit der Klasse "AcceptanceCode").
Code	java	Speichert den Code in einem String. Verwendet für Testcode und Klassencode.
AcceptanceCode	java	Erweitert die Klasse "Code" und speichert den Akzeptanzcode in einem String. Speichert außerdem, ob der Akzeptanzcode bestanden wurde (wird für die graphische Oberfläche benutzt).
State	java	Enum, repräsentiert die Phasen des (A)TDD-Zyklus, in der sich der Benutzer befinden kann.
CatalogRepository	java	Lädt und speichert den aktuell ausgewählten Katalog mit Hilfe von den Klassen "CatalogLoader" und "ConfigLoader". Beim Laden wird dem Benutzer ein Fenster zum Auswählen der Datei angezeigt.

AttdModel	java	Model-Klasse für die Hauptansicht der Programms, die die Hauptlogik des Programms umsetzt. Speichert und verwaltet, in welcher Phase des TDD-Zyklus sich der Benutzer momentan befindet, wie viel Zeit er noch bei einer möglichen Baby-Steps-Aufgabe hat, den Code der Aufgabe, die er gerade bearbeitet, und welche Fehler beim Kompilieren aufgetreten sind. Funktionsweise des Models siehe die Klasse "FxmlLoader".
AttdController	java	Controller-Klasse für die Hauptansicht des Programms. Siehe die Klassen "AttdModel" und "FxmlLoader".
CustomListView	java	Erweitert die Klasse "ListView" von JavaFX und ist dafür zuständig, die Aufgaben eines Katalogs anzuzeigen und auf die Mauseingabe des Benutzer zu reagieren.
CustomCallback	java	Hilfsklasse für die Klasse "CustomListView", ermöglicht es, ein Callback mit zwei Parametern aufzurufen.
ErrorControl	java	Erweitert die Klasse "TableView" von JavaFX und ist dafür zuständig, die Fehler anzuzeigen, die beim Kompilieren aufgetreten sind.
JavaStringCompiler Wrapper	java	Verbindet die Compiler-Bibliothek mit dem Programm. Wird verwendet, um den Code des Benutzers zu kompilieren. Dabei wird überprüft, ob die Testklasse einen Test enthält, oder ob die Testklasse nur aus dem Grund nicht kompiliert, dass auf noch nicht implementierte Felder/Methoden in der Codeklasse zugegriffen wird.
CustomCompiler Error	java	Erweitert die Klasse "CompilerError", wird von der Klasse "JavaStringCompilerWrapper" verwendet.
CustomTestResult	java	Erweitert die Klasse "TestResult", wird von der Klasse "JavaStringCompilerWrapper" verwendet.

mvc

FxmlLoader	java	Klasse, die die Umsetzung des Model-View-Controller Prinzips vereinfacht. Es können eine Klasse für den Controller, eine Klasse für das Model und ein Dateiname der View im fxml-Format übergeben werden. Dann werden neue Instanzen der beiden Klassen erstellt und die Instanz des Models wird automatisch in ein Feld der Instanz des Controllers geladen, das mit der Annotation "InjectModel" versehen wurde. Die fxml-Datei wird geladen und ihr wird der Controller zugewiesen. Das Ergebnis wird in einer Instanz der Klasse "ViewTuple" zurückgegeben.
ViewTuple	java	Wird von der Klasse "FxmlLoader" benutzt, um das Ergebnis zurückzugeben. Beinhaltet die Instanzen des Controllers, des Models und der View.
InjectModel	java	Annotation, die von der Klasse "FxmlLoader" benutzt wird, um zu bestimmen, in welches Feld des Models der Controller geladen werden soll.

workspace

WorkspaceController	java	Controller-Klasse für den Auswahl-Dialog des Workspace. Wird in der Klasse "AttdModel" benutzt, wenn der Benutzer den Workspace ändern will. Funktionsweise siehe die Klasse "FxmlLoader".
WorkspaceModel	java	Model-Klasse für den Auswahl-Dialog des Workspace. Siehe die Klassen "WorkspaceController" und "FxmlLoader".

loader

Config	java	Speichert den Pfad zum "Workspace-Ordner". Sobald der Benutzer zum ersten Mal eine Datei lädt oder speichert, wird dieser Pfad abgefragt. In diesem Ordner werden die Dateien und der Code des Benutzers gespeichert.
XmlLoader	java	Oberklasse von den Klassen "ConfigLoader" und "CatalogLoader", die Methoden zum Lesen und Schreiben von XML-Dateien zur Verfügung stellt.
ConfigLoader	java	Erweitert die Klasse "XmlLoader" und liest und speichert die “config.xml"-Datei, in der die Informationen einer Instanz der Klasse "Config" gespeichert werden. Sie liegt immer im Arbeitsverzeichnis des Programms, wenn sie nicht vorhanden ist, wird sie beim Speichern erstellt.
CatalogLoader	java	Erweitert die Klasse "XmlLoader" und liest und speichert eine Instanz der Klasse "Catalog" in einer angegebenen XML-Datei.

resources

AttdView	fxml	View-Datei für die Hauptansicht. Siehe die Klassen "AttdModel", "AttdController" und "FxmlLoader".
WorkspaceView	fxml	View-Datei für den Auswahl-Dialog des Workspace. Siehe die Klassen “WorkspaceController", "WorkspaceModel" und "FxmlLoader".
ListView	css	CSS-Datei, sorgt dafür, die Abstände der Einträge innerhalb einer "CustomListView" zu formatieren.