Systembeschreibung Projekt 7 - Gruppe: ZeNaMiKe

1. Vorhandene Packages, deren Klassen und ihre Hauptfunktionalitäten

Package: *compiler*

Compiler
Die Klasse Compiler ist wie der Name schon sagt für das Kompilieren des eingegebenen Codes verantwortlich. Die relevanten Informationen werden vom CompilerManger bereit gestellt, dann wird ein Testcompiler initialisiert und die Ergebnisse, die dieser liefert werden wiederum an ein erstelltes Objekt der Klasse CompileResults weitergegeben. Dieses gibt daraufhin eine Liste mit den Ergebnissen des Compilevorganges zurück.Die Verarbeitung des Codes und der Tests erfolgt zudem separat voneinander.
Collaborators: CompileResults, CompilerManager

CompilerManager
Die Klasse CompilerManager erstellt zunächst ein Objekt der Klasse Compiler, nimmt die in der GUI eingegebenen Informationen bezüglich Code und Test und ruft mit diesen jeweils den Compiler auf. Der Compiler gibt dann die Ergebnisse des Compilevorganges zurück und diese werden mit einem dem Durchgang entsprechenden Index in eine Liste mit allen Ergebnissen integriert. Auch hierbei werden Tests und Code separat voneinander verarbeitet.
Collaborators: Compiler, TDDController, Tracker

CompileResults
Die Klasse CompileResults verarbeitet beziehungsweise speichert die Informationen bezüglich der Ergebnisse des Compileprozesses. Diese werden in Form einer Liste vom Typen String gespeichert, die über den Compiler und im nächsten Schritt über den CompilerManager an die GUI zurückgeliefert wurde, wo die verarbeitet werden kann.
Collaborators: CompileManager

→ Hinweis: Alle Klassen dieses Packages interagieren mit der vk.core.api, die von Jens Bedisposto zusammen mit der Aufgabenstellung bereitgestellt wurde.

Package: *contentmanager*

TextManager

Die Klasse TextManager ist im Grunde dafür da, Eingaben die über die GUI erfolgen im Bezug auf die Settings, also Tracking und Babysteps, zu verarbeiten. Hierbei ist die Klasse in der Lage sowohl mit einer TextArea als auch mit einem String zu arbeiten, um die Verwendbarkeit der Klasse im System zu erhöhen und unnötige Redundanzen zu vermeiden.
Collaborators: TDDController, InformationCore, PluginManager

Package: *controller*

StartController
Die Klasse StartController verarbeitet die Buttons Setting, TaskList und TDDCycle, die im gleichnamigen FXML-File (*resources*) definiert werden. Bei Setting wird der SettingsLoader aufgerufen, bei TaskList der TaskListLoader und bei TDDCycle der TDDLoader.
Collaborators: StartLoader, SettingsLoader,TaskListLoader, TDDLoader FMXL: StartController.fxml

TaskListController
Die Klasse TaskListController verarbeitet die Buttons Back, TDDCycle und Go, die im gleichnamigen FXML-File (*resources*) definiert werden. Des Weiteren verarbeitet er den absoluten Pfad zu einem Aufgabenkatalog in dem Textfield und fügt die jeweiligen Aufgaben in die Choicebox ein, vorausgesetzt das das Format des Aufgabenkatalogs korrekt ist. Die Textarea zeigt dann, wenn eine Aufgabe ausgewählt wurde, die entsprechende Beschreibung an.
Collaborators: TaskListLoader, StartLoader, TDDLoader, CatalogManager FXML: TaskListController.fxml

SettingsController
Die Klasse SettingsController verarbeitet die Buttons Back und TDDCycle, die im gleichnamigen FXML-File (*resources*) definiert werden. Des Weiteren verarbeitet die Klasse zwei Checkboxen für die beiden Settings Tracking und Babysteps und speichert jeweils die Information ob das Setting aktiviert oder deaktiviert ist.
Collaborators: SettingsLoader, StartLoader, TDDLoader FXML: SettingsController.fxml

TDDController
Die Klasse TDDController verarbeitet die Buttons Back und Compile, die im gleichnamigen FXML-File (*resources*) definiert werden. Des Weiteren verarbeitet die Klasse die beiden Textareas und falls der Controller von dem TaskListLoader aus aufgerufen wird, werden die Areas mit den entsprechenden Code- und Testvorlagen gefüllt.
Collaborators: TDDLoader, CycleEnum, Cycle, StartLoader FXML: TDDController.fxml

Package: *loader*

StartLoader
Die Klasse StartLoader lädt die im entsprechenden FXML-File (*resources*) definierten Informationen, erstellt eine Scene mit diesen und setzt die Stage auf eben diese Scene.
Collaborators: StartController, TaskListController, SettingsController, TDDController,MainControlClass FXML: StartController.fxml

SettingsLoader
Die Klasse SettingsLoader lädt die im entsprechenden FXML-File (*resources*) definierten Informationen, erstellt eine Scene mit diesen und setzt die Stage auf eben diese Scene.
Collaborators: SettingsController, StartController,MainControlClass FXML: SettingsController.fxml

TaskListLoader
Die Klasse TaskListLoader lädt die im entsprechenden FXML-File (*resources*) definierten Informationen, erstellt eine Scene mit diesen und setzt die Stage auf eben diese Scene.
Collaborators: TaskListController, StartController,MainControlClass FXML: TaskListController.fxml

TDDLoader
Die Klasse TDDLoader lädt die im entsprechenden FXML-File (*resources*) definierten Informationen, erstellt eine Scene mit diesen und setzt die Stage auf eben diese Scene.
Collaborators: TDDController, StartController, TaskListController, SettingsController, MainControlClass FXML: TDDController.fxml

Package: *main*

Main
Die Klasse Main ist die Hauptklasse unseres Programmes, mit der eben dieses überhaupt erst gestartet werden kann. Sie enthält die überschriebene Start-Methode, in der eine neue Instanz der MainControlClass erstellt wird, mit der in Main erstellten Stage als Parameter.
Collaborators: MainControlClass

MainControlClass

Die Klasse MainControlClass wird von Main aufgerufen und initialisiert alle Loader Klassen, holt die jeweiligen Controller ein und lädt die entsprechenden Scenes aus den Loadern. Die Stage wird zunächst auf die Scene, die vom StartLoader geladen wird, gesetzt.
Collaborators: StartLoader, SettingsLoader, TaskListLoader, TDDLoader, Main

Package: *informationcore*

InformationCore
Die Klasse InformationCore ist neben TextManager eine der Knotenpunkte von Informationen. Diese werden in Form von Managern in InformationCore komprimiert, um eine Interaktion zu ermöglichen.
Collaborators: TextManager, CycleManager, SettingsManager

Package: *util*

⇒ **Subpackage***: manager*

CatalogManager
Die Klasse CatalogManager verarbeitet die Informationen, die man mithilfe von XMLCatalogReader aus einem gegebenen File ausliest. Er bereitet diese im Weiteren auf, damit diese vom TaskListController in die GUI eingebunden werden können.
Collaborators: TaskListController, XMLCatalogReader, Exercise

⇒ **Subpackage***: xml/reader*

Exercise
Die Klasse Exercise repräsentiert eine Aufgabe aus dem Aufgabenkatalog. Ein Objekt dieser Klasse hat einen Namen, eine Beschreibung, Beispielcode für die Tests und ein Grundgerüst für die Klasse. Alle diese Informationen werden in Form eines Strings gespeichert.
Collaborators: CatalogManager

XMLCatalogReader
Die Klasse XMLCatalogReader nimmt einen absoluten Pfad entgegen und liest das entsprechende File aus und speichert, falls es dem Format entspricht, die jeweiligen Aufgaben in Exercise-Objekten ab und sammelt diese in einer Liste.
Collaborators: Exercise, CatalogManager

XMLReader (**usage not decided**)

Die Klasse XMLReader liest ein gegebenes File aus, je nachdem ob das File dem Template der XMLWriter Klasse entspricht, wurden die Tags im XML ausgelesen oder es wurde nur ein String angefertigt mit dem Inhalt des Files.
Collaborators: -

⇒ **Subpackage***: xml/writer*

XMLWriter (**usage not decided**)
Die Klasse XMLWriter schreibt bzw. überschreibt ein gegebenes File und formatiert es nach einem vorher definierten Muster.
Collaborators: -

⇒ **Subpackage***: classnameparser*

ClassNameParser
Die Klasse ClassNameParser filtert den Namen einer Klasse aus einem Source Code, der in Form eines Strings übergeben werden sollte.
Collaborators: -

Package: *settings*

Setting
Die Klasse Setting repräsentiert ein einzelnes Setting.
Collaborators: Settings

Settings
Die Klasse Settings ermöglicht die parallele Verwaltung mehrerer Settings.
Collaborators: Setting, SettingManager

SettingsManager
Die Klasse SettingsManager ist für die parallele Verwaltung mehrerer Settings verantwortlich.
Collaborators: Settings

SettingException

Die Klasse SettingException wirft Exceptions, wenn ein ungewollter Zustand in den Settings erreicht wird.
Collaborators: SettingManager, Settings

Package: *plugin*

Plugin
Die Klasse Plugin ist eine Schnittstelle für Plugins.
Collaborators: Babysteps, Tracker

PluginManager
Die Klasse PluginManager ist eine Schnittstelle um Informationen des Programms and die Plugins weiterzureichen.
Collaborators: Babysteps

PluginLoader
Die Klasse PluginLoader macht es einfacherPlugins zu laden. Grund für die Existenz dieser Klasse ist die GUI.
Collaborators:

⇒ **Subpackage***: babysteps*

Babysteps
Die Klasse Babysteps ist dafür verantwortlich, dass nach einer gegeben Zeit der Text resettet wird.
Collaborators: PluginManager, Setting, Plugin

⇒ **Subpackage***: tracker*

Tracker
Die Klasse Tracker nimmt Informationen vom CompileManager und gibt diese visuell aus.
Collaborators: CompileManager, Plugin

TrackerController
Die Klasse TrackerController wird für die GUI-Realisierung verwendet.
Collaborators:

TimeCalculator
Die Klasse TimeCalculator erstellt ein Objekt vom Typ LocalTime und definiert die Methoden startTime() und endTime() um Zeitmessung zu beginnen und zu stoppen.
Collaborators:

PhaseInformation
Die Klasse PhaseInformation interagiert mit dem CompileManager und holt Informationen über die aktuelle Phase und die benötigte Zeit ein.
Collaborators: CompileManager

Package: *cycle*

Cycle
Die Klasse Cycle ist der Ort im Code, an dem der ganze Cycle-Prozess definiert wird. Sie handelt zudem andere Objekte und hängt stark mit der GUI zusammen.
Collaborators: CycleEnum, TDDController

CycleEnum
Die Klasse CycleEnum definiert nur die drei Phasen des TDD-Cycle.
Collaborators: Cycle, TDDController

CycleManager
Die Klasse CycleManager handelt den Cycle an sich, initialisiert die Cycles und holt Informationen über den aktuellen Cycle ein. Die Klasse ist zudem dafür verantwortlich, das die nächste Phase im Cycle eingeleitet wird
Collaborators: Cycle, CycleEnum, TDDController

Package: *controllerloader* (**usage unknown**)

Verwendete Quellen:

http://www.w3schools.com/css/; zuletzt aufgerufen am 15.Juli 2016
http://fxexperience.com/2011/12/styling-fx-buttons-with-css/; zuletzt aufgerufen am 15. Juli 2016
http://code.makery.ch/library/javafx-2-tutorial/; zuletzt aufgerufen am 14. Juli 2016
https://www.youtube.com/watch?v=K7BOH-Ll8_g&feature=youtu.be; zuletzt aufgerufen am 14. Juli 2016
https://www.youtube.com/watch?v=YWEhhyXI_hE; zuletzt aufgerufen am 14. Juli 2016

http://blog.buildpath.de/fxml-composition-how-to-get-the-controller-of-an-included-fxml-view-nested-controllers/; zuletzt aufgerufen am 14. Juli 2016

http://javarevisited.blogspot.de/2015/07/how-to-read-xml-file-as-string-in-java-example.html; zuletzt aufgerufen am 10. Juli 2016
http://www.java-samples.com/showtutorial.php?tutorialid=152; zuletzt aufgerufen am 11. Juli 2016

https://wiki.byte-welt.net/wiki/Java-Programme_durch_PlugIns_erweitern; zuletzt aufgerufen am 23.Juni 2016