Systembeschreibung des Abschlussprojektes

Inhaltsverzeichnis:

1.) Grundkonzept bzw. Idee
Die grundsätzliche Idee des TDDT liegt darin, Erstsemester in das Konzept der testgetriebenen Entwicklung (test-driven development, TDD) einzuführen.
Die Testgetrieben Entwicklung ist eine Strategie zur Entwicklung von Software, bei der immer nach dem gleichen Schema vorgegangen wird. Zunächst wird ein fehlschlagender Unit-Test geschrieben. Danach wird eine Methode geschrieben, die den fehlschlagenden Test erfüllt. Anschließend wird der Quelltext des Programms verbessert bzw. optimiert. Dieser Zyklus wird anschließend solange wiederholt, bis das Programm fertig ist.

2.) Umsetzung
Zuallererst haben wir uns Gedanken darüber gemacht was wir als User wollen. Da kommen schnell solche Worte wie „Nutzerfreundlich“ , „einfaches Designe“ und „leicht zu verstehen“ ins Spiel. Wir einigten uns schon während des ersten Gruppentreffens eine minimalistische Oberfläche zu bauen, ohne unnötige Funktionen Buttons und Regler. Ganz besonders wichtig war uns dagegen die Möglichkeit gleichzeitig Aufgabenstellung, Konsole, Tests und Code einzusehen. Somit wollen wir sicherstellen das in unserem Programm der User so effektiv wie möglich arbeiten.

3.) Feateures
Besonders möchten wir auf die beiden Extras aufmerksam machen: Einen Importeur und einen Exporteur. Der Importeur kann neue Aufgaben von einer Online Ressource laden, der Exporteur packt ein abgabefertiges Zip.

Daneben gibt es die beiden Features Babysteps und einen Akzeptanztest für ganze Features, deren Funktion ist ja bereits in der Aufgabenstellung beschrieben.

4.) Code:
Unser Programm baut auf teil unabhängigen Modulen auf. Dazu wurden mehrere Packages erstellt, die sich mit jeweils einer Kernaufgabe befassten bzw. typzusammengehörigen Aufgaben verwaltet. Ich werde nun ausführliche auf die einzelnen Packages eingehen.
Zunächst haben wir das Package „GUI\Handling“, das die Main App-Class enthält, in welcher sich die zur Ausführung der später automatisch generierten .jar-Datei notwendige Main-Methode befindet. Main App hat die Aufgabe, das Programmfenster aufzubauen.
Das Package „Compiler Handling“ enthält die Routinen, um sowohl die Tests als auch den Programmquelltext kompilieren zu können. Diese Routinen werden aus der zur Verfügung gestellten Compiler-Bibliothek „virtual-kata-lib“ bezogen, worauf ich später noch etwas ausführlicher eingehen werden. Abgefangen und gemeldet werden alle Arten von Errors, wohl Syntax/Compiletime-Errors als auch Run Time-Errors. Zudem besitzt die Klasse „Tester“ des Packages Methoden, die die Anzahl an fehlschlagenden, durchlaufenden und ignorierten Tests sowie die Zeit, die zum Compilieren und Testen gebraucht wird, zurückliefert.
Das Package „File Handling“ verwaltet mehrere Klassen, die für den Import/Export der im Zuge der Entwicklung entstehenden Dateien zuständig sind. Die Klasse FolderManager verwaltet die benötigte Ordnerstruktur, die unter anderem die Aufgaben beinhaltet.
Das Package „PopUpHandling“ behandelt die verschiedenen Meldungen bzw. die verschiedenen Alerts.
Das Package „SceneHandling“ beinhaltet die Schnittstelle zwischen allen anderen Paketen. Die Klasse Screen verwaltet unter anderem die GUI an sich, die Eingabemöglichkeiten, die

Mitteilungen und die Ausgaben, unter anderem der Compilermeldungen

Das Package „StringHandling“ enthält die Klasse StringFilter, die Strings kürzen und Namen für die zu erstellenden Klassen entwerfen kann und so für syntaktische Gleichheit bei den Namen sorgt.
Das Package „TabsHandling“ enthält Klassen, die die Tabs für Projekte darstellen, die in unserem Programm geschrieben werden und die Hierarchie des Projekts angeben.
Das Package „xmlHandling“ beinhaltet die Klasse XMLReader, die XML-Dateien, welche die Aufgaben enthalten, lesen kann und die in ihnen angegebenen Daten extrahiert, damit sie vom Programm weiterverarbeitet werden können.

5.) Lizenz
Wir haben uns für die MIT-Lizenz entschieden. Wir wollen unsere Software allen kostenfrei zur Verfügung stellen, aber auch als Ersteller des Rohentwurfs genannt werden.

6.) Abhängigkeiten und Gradle-Build
Das Programm ist nur von zwei externen Bibliotheken abhängig, nämlich von der Junit-Bibliothek für die Unit-Tests und der „virtual-kata-lib“, die die Compilerbibliothek ist. Die Abhängigkeiten werden während des Build-Prozesses automatisch von der Build-Software Gradle aufgelöst, die als Repository die Apache Maven-Bibliothek mavenCentral übergeben bekommt. Die Compilerbibliothek wird zur Compilezeit des Hauptprogramms benötigt, die Junit-Bibliothek wird zur Compilezeit der Unit-Tests benötigt, daher die Formulierung unter dependencies(). Als plugin wurde java als Sprache festgelegt. Zudem wurde im Build-Skript festgelegt, dass sich die main-Methode, die beim Ausführen der generierten -jar-Datei im Package „GUI\Handling“ in der Klasse MainApp befindet.