Systembeschreibung Projekt 7

Unser Programm nutzt die vorgegebene Bibliothek. Wir haben und für die Zusätze Babysteps und ATDD entschieden. Babysteps sorgt dafür, dass die Schritte nach einer gewissen Zeit zurückgesetzt werden (Anfangs wird noch geprüft, ob der Wert von Babysteps zwischen 2-3 Minuten liegt, ansonsten wir dieser auf 2 Minuten gesetzt). ATDD haben wir rechts oben eingebaut. Es kann ein Beispieltest geladen werden, um dem User klarzumachen was zu tun ist. Wenn der erste Test läuft wird RED freigegeben, und ab da kann dann am Programm oder am Test gebastelt werden. Eine genauere Beschreibung findet man in der Anleitung. Wichtig ist, dies ist ein ganz von sich auf aufgebautes Programm, d.h der User hat nur einen festen Weg seine Knöpfe zu drücken, sodass wir immer einen Zyklus unseres (A)TDD machen. (Anfangs geht halt nur der Übung reinladen Knopf)

Das Programm beinhaltet folgende Main Klassen:

Main Klassen:

- **Start:** Die Startklasse baut das Startfenster auf. Es werden alle Buttons, Labels oder Textfelder in die Maske geladen. Eine Übung aus dem Katalog kann nun durch den "Übung reinladen"-Button reingeladen werden. Dadurch kann noch auf der Festplatte die „Aufgaben.xml" gesucht und ausgewählt werden, da die Übungsgruppenleiter auch Updates der Aufgaben verschicken können. Durch die verschiedenen „Action"-Klassen werden dann die Buttons gesteuert und mit Funktionen versehen.

- **AkzepTestAction:** Diese Klasse führt den Akzeptanztest aus und gibt, bei erfolgreichem Test den RED Button frei. In der Maske heißt dieser Button: "Akzeptanztest starten um RED freizugeben".

- **BacktoRedAction:** Wenn man im Textfeld von „Green" ist kommt man mit diesem Button zurück ins „Red"-Textfeld. In der Anwendung heißt dieser Button: „Wechsle zu RED".

- **CaAction:** Die Klasse lädt einen funktionierenden Akzeptanztest ins Fenster. Dieser kann testweise verwendet werden. Eigentlich sollte in diesem Fenster ein eigener Akzeptanztest geschrieben werden. In der Maske heißt der Button: „Lade Akzeptanztest (nur fuer Testzwecke)".

- **CheckboxAction:** Wenn man im „Green"-Textfeld fertig ist kann man „Prüfe Programm" drücken. Man kann zusätzlich diese Checkbox („Check Akzeptanz") aktivieren um einen zusätzlichen Akzeptanztest zu durchlaufen.

- **PruefeProgActions:** Falls der Test erfolgreich ist wird dies in der Konsole ausgegeben, sonst muss der Test oder das Programm noch weiter abgeändert werden. Anschließend kann die Akzeptanztest Checkbox ausgewählt werden. In der Maske heißt der Button: „Pruefe Programm".

- **ExitSpeichernActions:** Am Ende aller Tests können die Tests / Programme gespeichert werden oder das Programm durch „exit“ beendet werden. Speichern mittels Button: „Dateien speichern“.

- **GreenAction:** Wenn „RED“ fehlschlägt wird „Green“ freigegeben und durch den Button das Textfeld aktiviert.

- **RedAction:** Sobald „RED“ freigegeben ist wird und der Button benutzt wird, wird hier das Textfeld aktiviert. Hier kann dann der Test bearbeitet werden.

- **StartTestAction:** Neben dem Textfeld von RED gibt es den Button „Starte Test“. Falls dieser bereits erfolgreich durchläuft wird kein weiteres Feld / kein weiterer Button aktiviert. Es wird ausgegeben, dass der Test erfolgreich war.

- **Speichern:** Schreibt mittels FileWriter eine neue Java Datei. Speichert also den Test oder die Funktion ab.

- **ReinladenClasse:** Ist dafür zuständig, dass eine Aufgabe aus dem Aufgabenkatalog reingeladen wird.

- **Positions:** Diese Klasse ist für die genauen Positionen der Buttons, Textfelder usw. im Startfenster zuständig.

- **LesenAusXml:** Wie der Name schon sagt, ist diese Klasse dafür zuständig bestimmte Aufgaben aus einer XML Datei auszulesen, die dann in unserer Maske geprüft und bearbeitet werden können.

- **JavaFile:** Ist eine Klasse die genutzt werden kann um neue Java Dateien zu erstellen. Zum Beispiel die geschriebenen Funktionen und Tests.

- **Compilieren:** Die Klasse prüft, ob die Funktionen durch den Test erfolgreich laufen oder nicht, und gibt eine entsprechende Konsolenausgabe aus.