Funktionsweise

Erste Konfiguration

Am Ende muss folgende Applikation erstellt werden: Ein User wählt eine Aufgabe aus dem Katalog aus, die gewisse **Auszeichnungs-Tags** *(in 'Projekt7.pdf wird XML vorgeschlagen)* enthalten.
Diese werden erkannt und bereiten dann ein **Projekt-Ordner** vor.

Sollte also zum Beispiel [Class] ....[/Class] in der Aufgabendatei vorliegen, wird dann erkannt, dass es sich um eine vordefinierte Klasse handelt.

(Siehe unter Projekt7.pdf **A Beispielkonfiguration**)

Es wird gefragt, ob **Baby-Steps** aktiviert werden soll.

Projekt-Ordner

Im Projekt-Ordner befinden sich drei Dateien:

Test.java
code.java
temp_code.java

Die aus dem Katalog geladene Aufgabe definiert die Daten (zum Beispiel durch XML) vor.

Das Programm arbeitet also nun fest im Projekt-Ordner.

Hauptfunktion

Zunächst befinden wir uns in **Status_0** [RED]:

Die aus dem Projekt-Ordner geladene Datei *test.java* wird geöffnet und kann bearbeitet werden. Sobald aus dem Projekt-Ordner vorhandene Datei *code.java* nicht kompiliert, oder der Test fehlschlägt, können wir zum nächsten **Status** wechseln.

Wir befinden uns nun in **Status_1** [GREEN]:

Wir können jetzt im Fenster den Code editieren. Es wird dazu *temp_code.java* geöffnet, eine Kopie von *code.java*.
Falls der User zurück zu Test [RED] wechselt, ohne, auf **Refactoring** gedrückt zu haben, wird *temp_code.java* wieder den Code aus *code.java* enthalten.

Sobald die in der Datei *test.java* ausgeführten Tests funktionieren, kann der User '**Refactoring**' klicken.

Wir befinden uns nun in **Status_2** [BLACK]:
Es wird *temp_code.java* in *code.java* umgelagert.
Der Code wird verbessert
Sobald **'Refactoring Finished'** geklickt wird, wird **Status_0** geöffnet.

(Seht dazu: 3 **Anwendungsbeschreibung: TDDT**)

Visualiserung der Idee

Um eine ungefähre Vorstellung zu bekommen, wie die App aussehen könnte.

Plan

Bis Ende nächster Woche soll das ganze als Konsolen-Applikation umgesetzt werden, bevor wir eine Schnittstelle zu JavaFX generieren. Nebenbei werden noch Skizzen zum Design erstellt.

Arbeitsteilung:

(wird Montag festgelegt)

Architektur

Nach gemeinsamer Überlegung aller Beteiligtne der Gruppe, haben wir uns folgende Architektur überlegt.

Klasse Datenbank{

Methode Node(){
Ordnerstruktur anlegen:
mit den JAV-Dateien; temp_code-java, test.java und code.java
Diese werden mithilfe der Aufgaben aus dem Katalog (mit Klassen, Aufgabenstellung etc.) vordefiniert.
}

Methode Speichern(){
if(Status = = 0)
Speicher_Code();
if(Status = = 1)
Speicher_Test();
}

Methode Speicher_Code(){
Speicher Stream in JAV-Datei;
}

Methode Speicher_Test(){
Speicher Stream in JAV-Datei;
}

Methode öffnen(){

if (status = = 0)
öffne temp_code.java
if(status = = 1)
öffne test.java;

}

Klasse Controller {

öffnet Dateien (Verwaltet input und output aus Datenbank)
...Manager()

}

Klasse Manager{

Verwaltet Logik (Status, Levelcounter etc.)

}