Bericht

Klassenübersicht:

Klasse: Aufgabe.java
Die Klasse beinhaltet 2 Strings, den Namen der Aufgabe die später im Programm bearbeitet wird und den Inhalt der Aufgabe.
Die Klasse schreibt aus den Strings eine Java Datei.

Klasse: AlertWindow.java
Die Klasse erstellt ein Fenster welches eine Meldung anzeigen lässt. Das Fenster muss erst bestätigt werden bevor man im Programm weiterarbeiten darf.

Klasse: AufgabenWindow.java
Die Klasse erstellt ein Fenster mit 3 Buttons. In den Buttons stehen die Aufgabennamen für die Aufgaben die später im Programm bearbeitet werden sollen.
Wenn ein Button angeklickt wird, wird die Aufgabe zum Bearbeiten an „Window.java“ weitergegeben. Dort wird sie in die Klasse „SaveLoad.java“ übergeben und ins Programm geladen.

Klasse: Countdown.java
Die Klasse Countdown wird aufgerufen, wenn der Button „babysteps“ im Programm ausgewählt wird. Es wird ein Fenster erstellt in dem der Benutzer die Zeit für die Babysteps in einem Textfeld eingeben kann. Außerdem gibt es dort die Bedingung, dass die Eingabe über 0 und unter 180 (Sekunden) sein soll.

Klasse: Log.java
In der Klasse Log werden die Log-Dateien für das Tracking gespeichert.
Zunächst wird über die Methode „erstelleLog“ eine Datei „Log.txt“ erstellt.
Dort wird mit BufferedWriter eingetragen wann der User welche Arbeitsschritte erledigt hat bzw. wann der User auf „Nächste Phase“ geklickt hat.
In der ArrayListe „log2“ werden die Zeiten gespeichert bei denen der User auf „Nächste Phase“ geklickt hat.
Zwei Integer „counter“ und „counter2“ sorgen dafür, dass die richtige Zeit aus der ArrayListe „gezogen“ wird und dass angegeben wird welche Phase bearbeitet wurde.
In der Methode Fehlerlog wird eine extra Log-Datei „FehlerLog.txt“ erstellt in der alle Fehler des Users stehen.

Klasse: LogikTest.java
Ein Test für unser eigenes Programm. Die Tests fragen die Bedingungen zum Übergang der Phasen ab.

Klasse: SaveLoad.java
In der Klasse SaveLoad werden die Java Dateien, die in der Klasse „Aufgabe.java“ erstellt wurden, zum Bearbeiten geladen. In der Methode „speichern“ wird mit BufferedWriter von der „TextArea“ der Hauptklasse in eine Datei geschrieben.
(Hauptklasse ist Window.java)
Mit der Methode „laden“ wird mit einem Scanner der Inhalt einer Java-Datei in ein „TextArea“ geladen.

Klasse: TaskReader.java
Die Klasse TaskReader hat ein Array in dem 3 Aufgaben und 3 Test-Dateien, die aus einem Katalog gelesen wurden, speichert.

Klasse: Tester.java
In der Klasse Tester wird die Kata-Libary benutzt.
In der Methode „CompileClass“ wird überprüft ob eine übergebene Test-Datei und Aufgaben-Datei kompilierbar sind. Der Name und Inhalt der Dateien wird übergeben, so wie ein „boolean“ in dem steht ob es sich bei der Datei um eine Test-Datei handelt.
In der Methode „testTesten“ wird getestet ob es genau einen fehlschlagenden Test gibt. Wie zuvor werden Namen, Inhalt und boolean übergeben.
In der Methode „funktTesten“ wird geprüft, dass es keine fehlschlagenden Tests gibt.
Wenn ein Test fehlschlägt werden in den Strings „errors“ und „errors2“ die Fehlermeldungen des Compilers gespeichert.
Die Fehlermeldungen können mit den Methoden “getErrorString“ und „getError2String“ an eine andere Klasse weitergegeben werden.
Der String errors wird für die Klasse „Log“ benutzt.
Der String errors2 wird dem User direkt ausgeben, wenn es einen Compiler Fehler oder mehr als einen fehlschlagenden Test gibt.

Klasse: Tracking.java
Bei der Klasse Tracking handelt es sich um einen „Timer“ für die Klasse Log. Tracking wird in der Klasse Window erstellt und dann an Log weitergegeben. Der Timer zählt Sekunden nach „oben“.

Klasse: WindowTimer.java
Die Klasse WindowTimer ist für die „babysteps“ zuständig. In der „run“ Methode wird mit einer If-Abfrage ein Maximal-Wert angegeben bei dem der Timer gestoppt wird.
Der Maximal-Wert wird von der Klasse Countdown übergeben.
In Zeile 20 wird mit „Window.setAnzeige“ die aktuelle Zeit an Window weitergegeben, dort wird die Zeit in ein Textfeld eingetragen.

Hauptklasse: Window.java
GUI:
Als „Haupt“-Layout wird ein BorderPane benutzt.
Die GUI hat 7 Buttons.
2 Buttons um zwischen den Phasen zu wechseln.
Ein Button um den Katalog und ein Button um die Aufgaben auszuwählen.
Der Button „Phase“ gibt farblich die Phasen wieder und sagt dem User was zu tun ist.
Ein Button um die Babysteps zu aktivieren und ein Button um Die Log-Dateien zu erstellen.
Es gibt 2 TextAreas, im linken werden die Test-Dateien und rechts werden die Aufgaben-Dateien geladen. Je nachdem welche Phase gerade aktiviert ist, wird das TextArea ein- oder ausgeblendet.
Zusätzlich gibt es ein Textfeld welches die Zeit für die Babysteps anzeigt.
Logik:
Mit dem Button „katalog“ wählt man den Katalog mittels FileChooser aus. Dieser wird dann von „TaskReader“ übernommen und eingelesen und in entsprechende Java Dateien verarbeitet. Man kann nur einen Katalog aufrufen. Mit dem Button „aufgabe“ wird ein neues Fenster aufgerufen, indem man, wie in der Klasse „AufgabenWindow“ beschrieben, die

Aufgabe, die man bearbeiten möchte, auswählt. Die Test-Klasse wird dann in den linken Editor geladen. Man kann jederzeit die Aufgabe wechseln. Mit dem Button „babysteps“ wird ein neues Fenster aufgerufen, indem man die Zeitobergrenze eingibt. Dann wird ein neuer Timer aufgerufen, der dann bis zur entsprechenden Zeit hochzählt. Wenn vorher schon ein Timer existiert hat, wird dieser beendet. Wenn die Zeit abgelaufen ist wird je nachdem in welcher Phase und Editor man ist die entsprechende Klasse/Test neu geladen und somit das Eingegebene gelöscht. Wenn man auf den Button „nächste Phase“ klickt wird mit Hilfe der bool'schen Variablen „code“, „test“ und „refactor“ überprüft, in welcher Phase sich der Nutzer befindet. Je nachdem welche Phase aktiv ist, wird mit Hilfe der Klasse „Tester“ überprüft, ob die Bedingungen für einen Wechsel in die nächste Phase erfüllt ist. Wenn ja wird der eingegeben Text von dem zurzeit aktiven Editor mit Hilfe von „SafeLoad“ in der richtigen Java-Datei gespeichert. Außerdem wird dann noch für die nächste Phase der Test oder die Klasse in den jeweiligen Editor geladen (siehe Beschreibung GUI) und die Farbe und der Text des Buttons „phase“ wird an die nächste Phase angepasst. Wenn Babysteps aktiviert wurde beginnt bei jeder neuen Phase außer bei „Refactoring“ ein neuer Timer mit der zuletzt eingegebenen, maximalen Zeit. Man kann jederzeit diese Zeit ändern über den Button „babysteps“. Der Button „lPhase“ ist nur in der grünen Phase verfügbar, um vom Code zum Test schreiben zurück zu wechseln. Dabei wird der Button „phase“ wieder an die Phase „Grün“ angepasst, es wird zum linken Editor gewechselt und die Eingabe beim Code wird nicht gespeichert.

Aufbau des Kataloges:

Der Katalog muss wie folgt aufgebaut sein (Beispielkatalog: siehe Katalog.txt):

<katalog>
<klasse>
Name der 1. Klasse/Aufgabe
Inhalt der 1. Klasse/Aufgabe (kann mehrzeilig sein)
<klasseEnde>
<test>
Name des Testes für die 1. Klasse/Aufgabe
Inhalt des Testes für die 1. Klasse/Aufgabe (kann mehrzeilig sein)
<testEnde>
<klasse>
Name der 2. Klasse/Aufgabe
Inhalt der 2. Klasse/Aufgabe (kann mehrzeilig sein)
<klasseEnde>
<test>
Name des Testes für die 2. Klasse/Aufgabe
Inhalt des Testes für die 2. Klasse/Aufgabe (kann mehrzeilig sein)
<testEnde>
<klasse>
Name der 3. Klasse/Aufgabe
Inhalt der 3. Klasse/Aufgabe (kann mehrzeilig sein)
<klasseEnde>
<test>
Name des Testes für die 3. Klasse/Aufgabe
Inhalt des Testes für die 3. Klasse/Aufgabe (kann mehrzeilig sein)

<testEnde>
<katalogEnde>

Der Katalog muss immer aus 3 Aufgaben bestehen und hinter jeder Klasse/Aufgabe muss der entsprechende Test für diese Klasse sein.

Teambeteiligung:

Nur eine sehr grobe Einteilung, da eigentlich das meiste im Programm zusammen gemacht wurde und jeder jedem geholfen hat.

Hakan: GUI und grobe Struktur erstellt / Phase „Grün“ erstellt / Tracking / Babysteps/ Kata-Libary /

Robin Ahlers: Aufgaben und Katalog / Phase „Rot“ erstellt / „LogikTest.java“ / Babysteps / Kata-Libary /

Robin Schledz: TextArea in Datei und andersrum / Phase „Schwarz“ erstellt / Tracking / Babysteps / Gradle Travis /