Dokumentation

Javadoc: https://propra16.github.io/programmierpraktikum-abschlussprojekt-team-1/
Da die genaue Funktionsweise der Methoden hier erklärt wird, wird im folgenden Text darauf verzichtet, diese noch einmal zu erklären.

Zunächst haben wir eine interne Datenstruktur erstellt, in der ein gesamter Projekt gespeichert werden kann. In ein „Project“ werden der Name, eine Beschreibung, Babysteps (und deren Dauer falls sie genutzt werden), Tracking und eine Liste an Classen und Tests gespeichert (beide als Code-Objecke). Es gibt Methoden zum Testen und bearbeiten der enthaltenen Class- und Test-Objekte (durch Aufruf der Code-Methoden).
Das Interface Code wird von Class und Test implementiert die Methoden zum manipulieren und „getten“ des gespeicherten Codes enthalten und sich als CompilationUnit zurückgeben können.
Dieses Projekt wird, wenn es nötig ist um aktuelle Änderungen einzubinden, aktualisiert (mit den aus der GUI erhaltenen Daten).

Beim Start kann man auswählen ob man ein neues Projekt erstellen möchte oder ob man ein Projekt laden möchte. Dieses Fenster wird vom AlertHandler angezeigt.
Wenn ein Projekt neu erstellt werden soll wird eine neue „ProjectSettings“ Stage erstellt, in der Name, Beschreibung und Projekteinstellungen eingestellt werden können. Nun wird nach einem Klassennamen gefragt und mit diesem ein Class-Objekt erstellt, das der Classlist des aktuellen Projekts hinzugefügt wird. Danach wird das Hauptfenster, das in der Gui erstellt wird, angezeigt in diesem nun programmiert werden kann.
Soll ein bestehendes Projekt geladen werden, wird eine „Catalog“-Stage erstellt und die dazugehörige XML-Datei in den Katalog geladen. Das Laden der XML-Datei übernimmt dabei der „XMLHandler“.
Wird nun ein Projekt ausgewählt, wird das ausgewählte Projekt in das aktuelle „Constants“-Objekt gespeichert und es wird das Hauptfenster angezeigt.

In diesem Hauptfenster gibt es drei Tabs (Test, Code, Konsole), die jeweils eigene Klassen haben(„TestPane“, „CodePane“, „ConsolePane“). Zudem gibt es diverse Buttons (test, compile, next, back, save), die in der „Gui“ erstellt werden. Bei test, compile und next wird jeweils zunächst das Projekt geupdatet, indem die im Hauptfenster eingegebenen Daten in die interne Datenstruktur eingespeichert (genauer in den jeweiligen Test-, oder Class-Objekte). Anschließend werden bei „compile“ und „test“ die jeweiligen Funktionen in Project aufgerufen. Das Testen erfolgt über statische Funktionen der Testing-Klasse, der die CompilationUnits der zu testenden Klassen und Tests übergeben werden. Außerdem werden beim Testen/ Compilieren Fehler/ Rückmeldungen auf der Konsole ausgegeben.
Beim Auslösen des „save“-Buttons wird mithilfe des „XMLHandler“s das neue Projekt in eine XML-Datei gespeichert.
Beim Auslösen des „next“-Buttons wird die aktuelle Phase abgefragt (über das Phase-Objekt) und kontrolliert ob die jeweiligen Bedingungen für den Übergang in die nächste Phase erfüllt sind (über die Test-Methoden des Projects).
Ist dies der Fall wird in die nächste Phase übergegangen, indem die Phase im Phasen-Objekt um eins erhöht wird und alle nötigen Änderungen für die nächste Phase eingestellt werden (anzeigen der aktuellen Phase, disablen der Textareas die nicht bearbeitet werden sollen etc). In der Phase Refactor werden zusätzlich zu diesen Änderungen auch die Code-Objecte (Class und Test) aktualisiert, dh der aktuelle code wird als old_code abgespeichert, da dieser nun einmal die Tests bestanden hat und falls aktuelle Änderungen verworfen werden der alte Code wieder genommen werden kann.

Sind Babysteps aktiviert, so wird ein Timer-Objekt erstellt, dass von einer gegebenen Zahl herunterzählt und sobald 0 erreicht wird eine booleanProperty ändert. Dieser wird in „gui“ ein ChangeListener hinzugefügt. Ist also die Zeit abgelaufen wird die aktuelle Phase temporär gespeichert, ein Klick auf den next-Button simuliert und nun wieder die Phase abgefragt. War der Übergang in die nächste Phase nicht erfolgreich, so werden die letzten Änderungen verworfen.
Beim Übergang in die nächste Phase wird der Timer resettet, beginnt also erneut den Countdown.

Ist Tracking aktiviert wird ein EventHandler erstellt und Events können hinzugefügt werden. Beim Übergang in die nächste Phase wird ein PhaseStartEvent hinzugefügt. Ein PhaseStartEvent extendet Event und speichert somit die aktuelle Zeit beim erstellen eines PhaseStartEvents ein.

Wenn das Programm beendet wird öffnet sich, wenn Daten zur Erstellung einer Statistik da sind, eine StatisticStage die eine Statistik anzeigt.