Systembeschreibung

Behandlung offener Fragen:

Die Bearbeitungszeit für BabySteps wird in der Konfiguration einer Aufgabe in Sekunden angegeben. Dort wird auch angegeben, ob für eine Aufgabe die BabySteps-Funktion eingeschaltet ist.

Der Timer hat das Format „mm:ss“.

Klassen und Ihre Zuständigkeiten:

BabyStepsConfig.java:
Diese Klasse ist zuständig für die Initialisierung und Aktualisierung eines Timer-Labels oder anders ausgedrückt für all das, was keine Komponenten des Controllers benutzt. Mit der init()-Methode wird ein Label an eine Stringproperty (sp=Klassenvariable) gebunden und zu einer gegebenen Sekundenanzahl im Format „mm:ss“ initialisiert. Mit der count()-Methode wird der Timer dann runtergezählt. Dabei wird ein neuer Thread erstellt, damit parallel Aktionen möglich sind. Dieser wird eine Sekunde angehalten, der Timer um eine Sekunde reduziert und die aktuelle Zeit berechnet sowie über die StringProperty aktualisiert.

StoppUhr.java:
Diese Klasse stellt Methoden zur Verfügung, mithilfe derer man eine Startzeit sowie eine Endzeit setzen kann, sodass man mithilfe einer weiteren Methode die Zeit dazwischen messen kann.

Controller.java:
Diese Klasse enthält die gesamte Logik des Programms. Unter Anderem werden hier alle Aspekte zu „BabySteps“ abgearbeitet, die nicht das Timer-Label an sich betreffen, sondern vielmehr die Aktionen, die ausgeführt werden, wenn das Label einen bestimmten Wert hat. Die Methode „babyStepsHandling“ führt dabei die Option BabySteps der Logik nach aus, die Methode „babyStepsAbbruch“ dagegen prüft parallel, wann die Zeit abgelaufen ist und führt in Abhängigkeit davon entsprechende Aktionen aus.

Main.java:
Die Main-Klasse startet das Programm, setzt die Stage und lädt die FXML-Datei.

Interaktionen zwischen den Klassen:

Controller.java, BabyStepsConfig.java, Main.java:

Die Option „BabySteps“ wird im Controller durch die Methode „babyStepsHandling“ erfasst. Sie wird jeweils in der initialize-Methode und der next-Methode aufgerufen. Ausgeführt wird die Option, wenn in der Konfiguration einer Aufgabe BabySteps eingeschaltet ist und man sich nicht in der Refactor-Phase befindet (d.h. wenn cycle != REFACTOR). Sie ruft ihrerseits die Methoden „init“ zur Initialisierung des Timer-Labels sowie zur Bindung des Labels an die StringProperty sp und dann „count“ zum Runterzählen der Zeit in einem Neben-Thread auf. Dabei werden auch die Inhalte der TextAreas „testArea“ oder “codeArea“ in der Stringvariable Puffer zwischengespeichert, je nach dem in welcher Phase man sich befindet (RED oder GREEN).
Parallel dazu läuft die Methode „babyStepsAbbruch“, die in einem weiteren Thread prüft, ob die Zeit des Timer-Labels abgelaufen ist, also ob 0:00 erreicht wurde. Wenn dies der Fall ist, wird die für die aktuelle Phase maßgebende TextArea auf den im Puffer zwischengespeicherten Wert zurückgesetzt.
Die Boolean-Klassenvariablen successfullCompiling, refactoring und finished der Controller-Klasse sowie die Boolean-Klassenvariable stopThread sorgen dafür, dass die beiden Threads aus BabyStepsConfig.count() und Controller.babyStepsAbbruch() rechtzeitig abgebrochen werden, bspw. wenn zwischenzeitlich in die Refactoring-Phase gewechselt wurde oder wenn erfolgreich kompiliert wurde. StopThread sorgt dafür, dass frühere Threads abgebrochen werden. Finished sorgt dafür, dass der Thread in Controller.babyStepsAbbruch() stoppt, wenn der Thread in BabyStepsConfig.count abbricht, weil dann die Überprüfung der ablaufenden Zeit überflüssig wird. In der Main-Klasse wird über setOnCloseRequest sichergestellt, dass alle Threads stoppen, wenn das Fenster geschlossen wird.