Bericht – Projekt 7

Überblick

Das Programm startet mit der Main Methode, im Package Main. Der PhaseManager übernimmt die Hauptarbeit. Die GUI & GUI Catalog übernehmen die Visualisierung der Arbeit des PhaseManagers. Aktionen die synchronisiert werden müssen laufen über einen EventBus. Zur Unterstützung seiner arbeitet nutzt der PhaseManager CatalogDataSource & Exercise. Implementiert sind die Erweiterungen Tracking und Babysteps.

Beschreibung der Executer/PhaseManager

Zur Verwaltung der einzelnen Phasen wird ein „PhaseManager“ bereitgestellt. Dieser verwaltet ebenso den „TrackingManager“ und den „BabystepsManager“. Wenn der Benutzer den Button „next step“ klickt, wird die „checkPhase“ Methode des „PhaseManager“s aufgerufen. Diese bekommt die aktuelle „Exercise“, also das, was der Benutzer im Code- und im Testfeld stehen hat, übergeben, und überprüft, abhängig von der jeweiligen Phase, in der man sich gerade befindet, ob, falls dies gewünscht ist, angezeigt durch den boolean „continuePhase“, welcher ebenfalls mit übergeben wird, in die nächste Phase weitergegangen wird. Dazu wird ein Objekt vom Typ „Executor“ erstellt und auf diesem die Methode „execute“ aufgerufen, welcher wieder die „Exercise“ übergeben wird.
Der „Executor“ verwendet daraufhin die von Herrn Dr. Bendisposto bereitgestellte Bibliothek, um den Code und die Tests auf Kompilierfehler und Test Fehler zu überprüfen, wobei letzteres nur geschieht, wenn es keine Kompilierfehler gab. Das Ergebnis hiervon wird in einem Objekt vom Typ „ExecutionResult“ gespeichert, welches bei Kompilierfehlern das „CompilerResult“ (aus der Bibliothek) und eine Liste an Kompilierfehlern, ohne Kompilierfehler das „CompilerResult“ und das „TestResult“ (aus der Bibliothek) enthält.
Der „PhaseManager“ nutzt nun dieses „ExecutionResult“, um zu prüfen, ob dieses für die aktuelle Phase gültig ist, und dementsprechend startet er den „BabystepsManager“, wechselt die Phase und erstellt ein Objekt vom Typ „PhaseStatus“, welches enthält, ob die „Exercise“ gültig ist, das „ExecutionResult“ und die Phase, in welcher man sich ab jetzt befindet. Außerdem wird in jedem Fall die Methode „track“ des „TrackingManager“s aufgerufen und die „Exercise“, wie auch der „PhaseStatus“ übergeben. Dem „EditorViewController“ wird mithilfe des „EventBus“ ein Objekt vom Typ „ExecutionResultEvent“, welches den „PhaseStatus“ beinhaltet, gepostet und der „PhaseStatus“ zurückgegeben.
Im „PhaseManager“ kann man auch mit „resetPhase“ die aktuelle Phase zurücksetzen, wobei man immer in die Phase RED springt, außer man befindet sich in REFACTOR, dann wird eine Exception geworfen. Auch hier wird der „BabystepsManager“ wieder verwaltet und gestartet.
Mit „selectExercise“ wird aus dem bereitgestellten Katalog an Aufgaben eine solche ausgewählt, der „BabystepsManager“ wird wieder verwaltet, sowie der „TrackingManager“, und der GUI wird ein „ExerciseEvent“ gepostet, welches die ausgewählte „Exercise“ enthält.
Auch die Anzeige des Trackings läuft über den „PhaseManager“, welcher aber nur die Methode „displayInNewWindow“ des „TrackingManager“s aufruft.

Beschreibung der GUI/Events

Main Methode, startet GUI und instanziiert Phasenmanager, Trackingmanager sowie den Eventbus zur Kommunikation. Die Spracheinstellung wird per default auf Englisch gesetzt.

Der erste Output für den User ist ein RootLayout, der über den FXMLLoader bereitgestellt und mit der zugehörigen Steuerung RootLayoutController verknpüft wird.

Das Layout verfügt über ein Menü mit verschiedenen Unterpunkten/Items die per onAction mit den auszuführenden Methoden im Controller verbunden sind. Die Texte der Elemente sind mit Identifieren versehen. Diese werden aus der Ressourcen-Datei abhängig von der Spracheinstellung gesetzt. Buttons, Label zur Interaktion mit dem User werden per css Stylesheet formatiert.

Bei der Initialisierung des RootLayout wird gleich die EditorView mitgeladen. Hier sind alle Elemente zur Usereingabe des Codes/der Tests & Fehlerausgabe implementiert. Besonderheit sind die JavaCodeAreas, die die die Schlüsselwörter im JavaCode hervorheben.

Der EditorViewController enthält Methoden zur optischen Anpassung an die Phase changePhase (toGreen/Red /Refactor). Show Exercise Event, welches „@Subscribe“ notiert ist und auf ein update vom Eventbus horcht. Hier wird der Code/Test geladen. Ähnlich verhält sich showExecutionResult, welche abhängig vom Eventbus, die Ausgabe des Fehlercodes übernimmt und formatiert.

Beschreibung der Logik

Das Programm verwendet an vielen Stellen die Exercise Klasse.
Diese stellt eine Aufgabe aus dem Katalog dar, welche einen Titel, eine Aufgabenbeschreibung und zugehörige Skelett-Klassen für Code und Test enthält.
Zusätzlich wird in der Exercise Klasse gespeichert, ob die Baby Steps Erweiterung aktiviert ist und wie viel Zeit für das Schreiben der Tests und Programmteile zur Verfügung steht, bevor die Erweiterung die aktuelle Phase zurücksetzt.

Die Exercises werden über eine CatalogDatasource geladen.
Hier habe ich das CatalogDatasourceIF Interface erstellt.
Es gibt eine FakeCatalogDatasource implementierung, welche immer 3 hart verdratete Exercises zurück gibt. Die erste einthält in ihmen Code keine Fehler, die zweite hat einen Compile-Error und die dritte einen Test/Assertion-Error.

Die zweite Implementierung ist die XMLCatalogDatasource, welche die Exercises aus einer angegebenen XML Datei lädt. Die XML Datei wird mit den, von der JVM mitgelieferten javax.xml.* und org.w3c.dom.* Klassen geparsed.

Für die GUI gibt es einen ExerciseSelector, welcher eine CatalogDatasource übergeben bekommt und dann alle, im Katalog enthaltenen Exercises lädt. Die Titel der Exercises werden in einer Liste angezeigt und in einem Textfeld wird die zugehörige Beschreibung der markierten Exercise dargestellt. Beim Klick auf "Select" wird die ausgewählte Exercise zurückgegeben.

Beschreibung der Erweiterung Babysteps

Klassen: BabystepsManager

Aufgabe(n):

- enthält einen Konstruktor, übergibt den PhasenManager.
- start & stop Methoden. Die Start-Methode startet insofern noch gerade nicht bereits aktiv, einen Thread der prüft ob die Zeit für die Phase in dieser Übung abgelaufen ist.

- Enable & Disable Methoden

Abhängigkeiten

- PhaseManagerIF
- BabyStepsIF

Klassen: BabystepsManagerIF

Aufgabe(n):

- enthält einen Konstruktor, übergibt den PhasenManager.
- start & stop Methoden. Die Start-Methode startet insofern noch gerade nicht bereits aktiv, einen Thread der prüft ob die Zeit für die Phase in dieser Übung abgelaufen ist.
- Enable & Disable Methoden

Abhängigkeiten

- BabystepsManager

Beschreibung der Erweiterung Tracking

Der TrackingManager wird zu Beginn des Programms in der Main-Klasse instanziiert. Bei jedem Versuch des Users, in eine neue Phase zu wechseln (kein tatsächliches Wechseln notwendig!) wird vom PhaseManager die Methoder track des TrackingManagers aufgerufen.
Dieser Methode wird der aktuelle Stand der Exercise und der PhaseStatus übergeben. Die Methode instanziiert daraufhin einen Snapshot.
Das Objekt Snapshot ist wie eine Momentaufnahme des aktuellen Status der Bearbeitung zu verstehen. Es wird darin das Exercise-Objekt, der PhaseStatus und die aktuelle Uhrzeit gespeichert.
Dieses Snapshot-Objekt wird anschließend von der Methode track in einer Liste gespeichert.
Sollte der User auf den Verlauf-Button klicken wird über den PhaseManager die Methode displayInNewWindow aufgerufen. Hier wird die Tracking-FXML-Datei geladen, die PrimaryStage eingerichtet, der Titel des Fensters gesetzt und eine Mindesgröße des Fensters festgelegt.
Die Grundlage des Aufbaus bildet eine BorderPane. Außerdem wird die Methode generateTrackline des TrackingControllers aufgerufen.
In dieser wird ein neues Fenster geöffnet, in dem eine ähnliche Ansicht wie im Hauptfenster erzeugt wird (Zwei Felder für Code, darunter ein Feld für Ausgaben der Konsole). Dazu wird die EditorView-FXML-Datei geladen. Zusätzlich wird die Methode generateRectangles aufgerufen, die eine Trackline erzeugt, welche wiederum aus MyRectangle-Objekten besteht.
Die Klasse MyRectangle erbt von der Klasse Rectangle und bekommt beim Instanziieren zwei doubles (Höhe und Breite des Rechtecks) und ein Snapshot-Objekt übergeben. Die Größe des MyRechtangle-Objekts hängt von der Dauer ab, die der User in der zugehörigen Phase verbracht hat. Diese Zeit wird von der Methode getTimeBetweenSnaps des TrackingManagers berechnet. Die Farbe des Rectangles (rot,

grün, grau) hängt ebenfalls von der Phase ab und wird von der Methode getColor des Snapshot-Objekts ermittelt.
Damit repräsentiert jedes MyRectangle-Objekt einen Zeitpunkt im Verlauf der Aufgabe. Wenn der User eines der aneinandergereihten Rechtecke anklickt, wird in den Fenstern darunter genau der Code und die Konsolenausgabe angezeigt, die der User auch zu dem Zeitpunkt gesehen hat, als er mittels eines Klicks in die nächste Phase wechseln wollte.
Die reset-Methode des TrackingManagers setzt den Startpunkt der Aufgabe neu und löscht alle bisher gespeicherten Snapshots. Sie wird beispielsweise aufgerufen, wenn der User sich entscheidet, eine andere Aufgabe zu bearbeiten.