Die Hausaufgaben müssen von jedem Studierenden einzeln bearbeitet und abgegeben werden. Für die Hausaufgabe sind die aktuellen Informationen vom Blog https://seblog.cs.uni-kassel.de/ws2223/programming-and-modelling/ zu berücksichtigen.

Abgabefrist ist der 15.12.2022 - 23:59 Uhr

Abgabe

Wir benutzen für die Abgabe der Hausaufgaben Git. Jedes Repository ist nur für den Studierenden selbst sowie für die Betreuer und Korrektoren sichtbar.

Für die Hausaufgabe benötigst du **kein neues** Repository. Es wird das gleiche Repository benutzt, das bereits in Hausaufgabe 4 angelegt wurde. Dieses kann über folgenden Link erstellt werden, falls nicht bereits geschehen:

https://classroom.github.com/a/n-8XoDkO

Nicht oder zu spät gepushte (Teil-)Abgaben werden mit 0 Punkten bewertet!

Abgaben, die nicht lauffähig sind, werden mit 0 Punkten bewertet!

Projekte, deren GUI nicht mit FXML-Dateien umgesetzt sind, werden mit 0 Punkten bewertet!

Aufgabe 1 – Fxml (10P)

In dieser Aufgabe soll die Grundlage zur Umsetzung der in der vorherigen Hausaufgabe erstellten Wireframes geschaffen werden.

Vorbereitung

Wir verwenden für die Erstellung unserer Oberflächen den SceneBuilder. Lade dir den SceneBuilder unter folgendem Link herunter:

`https://gluonhq.com/products/scene-builder/`

Wenn du deine eigenen Wireframes nicht umsetzen möchtest, können die von uns zur Verfügung gestellten Wireframes (Abbildung 1 und 2) als Arbeitsgrundlage verwendet werden.

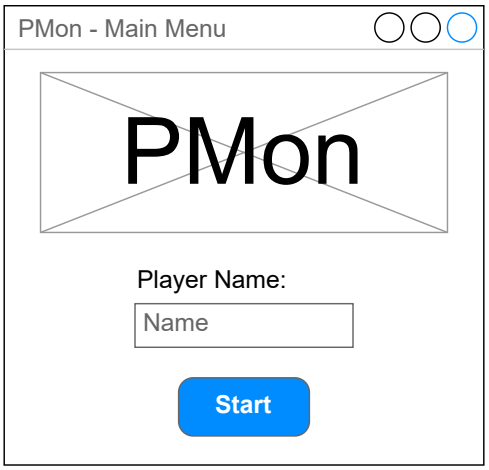

Abbildung 1: Login-Bildschirm

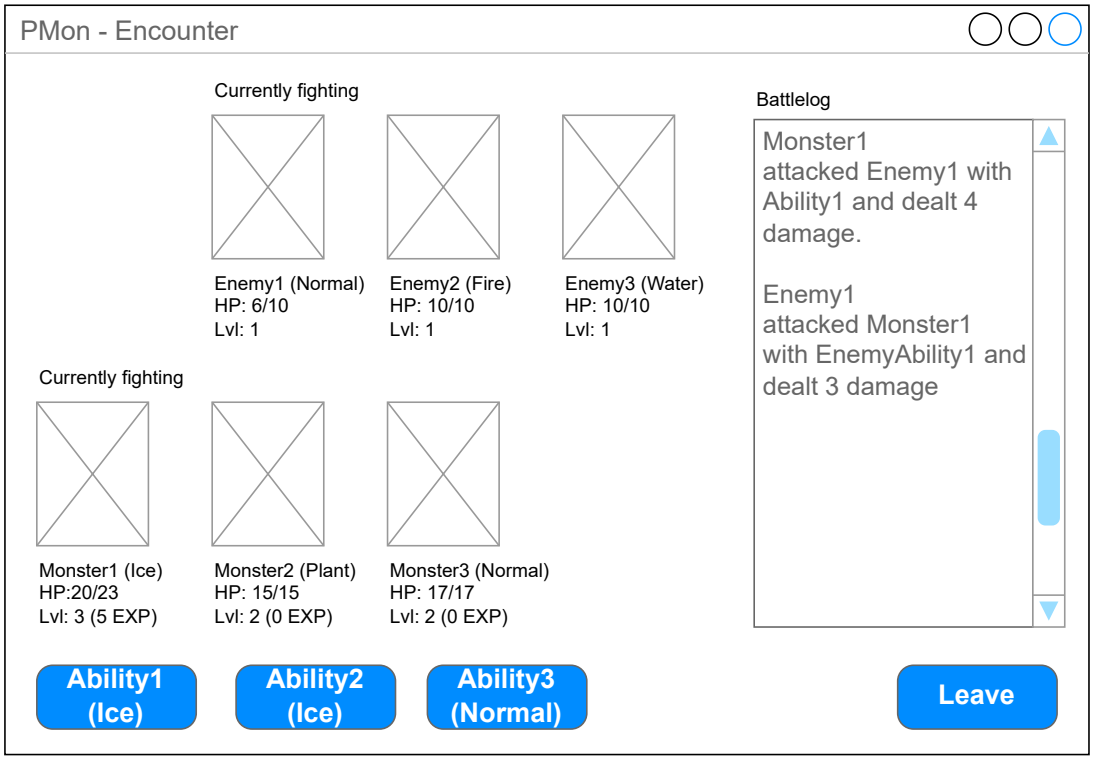

Abbildung 2: Battle-Bildschirm

1.1 Fxml-Dateien erstellen

Der Scenebuilder speichert seine Dateien im fxml-Format. Für jedes Wireframe muss eine eigene fxml-Datei erstellt werden (Login.fxml und Battle.fxml). Die zwei erstellten fxml-Dateien sind im Modul src/main/resources im Package de.uniks.pmws2223.view abzulegen. Achte darauf, dass im resources-Ordner die korrekt geschachtelte Ordnerstruktur des erforderten Packages erstellt wurde und nicht nur ein Ordner mit dem Namen „de.uniks.pmws2223.view“ existiert (dies ist leicht im File Explorer ersichtlich).

1.2 fx:ids vergeben

Sowohl beim Umsetzen deiner eigenen als auch unserer Wireframes sind **vorgegebene Benennungen** der fx:id bestimmter Komponenten zu berücksichtigen. Achte bei deinen eigenen Wireframes darauf, dass alle unten genannten Elemente vorhanden sind.

1.2.1 Login

Im Login-Bildschirm werden folgende fx:id vergeben:

- nameInput für die Text-Eingabe für den Spielernamen
- startButton für den Button, der das Encounter startet

1.2.2 Battle

Im Battle-Bildschirm werden folgende fx:id vergeben:

- encounterMonsterList für den Container, der die Monster des Encounters enthält
- playerMonsterList für den Container, der die Monster des Spielers enthält
- battleLog für den Textbereich, der den Kampfverlauf enthält
- abilityBar für den Container, der die Ability-Buttons enthält
- leaveButton für den Button zum Verlassen des Encounters

Die Blöcke, in denen Monster dargestellt werden, dürfen zunächst kopiert werden. Später werden wir diese in eine weitere Fxml-Datei auslagern. Beim Kopieren ist jedoch wichtig, dass die **fx:ids jeweils nur einmal vergeben werden**, andernfalls ist die Fxml-Datei fehlerhaft und kann nicht geladen werden. Kopiere daher **erst** deine Monster-Blöcke und vergib **danach** für den ersten davon die nachfolgenden fx:ids:

- nameLabel für den Text, der Name und Typ enthält
- hpLabel für den Text, der HP/MaxHp enthält
- lvlLabel für den Text, der Lvl und bei eigenen Monstern die Exp enthält

Committe und pushe die neuen Dateien abschließend auf den main-Branch.

Achte darauf, das Repository der aktuellen Hausaufgabe zu verwenden.

Aufgabe 2 – JavaFX (10P)

In dieser Hausaufgabe wird das Oberflächenframework JavaFX sowie die TestFX-Library verwendet.

2.1 Anpassungen der build.gradle

Um deinem Projekt die genannten Frameworks hinzuzufügen, muss die bestehende build.gradle erweitert werden. Dafür wird die build.gradle-Datei[1] bereitgestellt, die du in deinem Projekt ersetzen kannst.

2.2 Vorgegebene Klassen

Um einen einheitlichen Startpunkt zur Implementierung der JavaFx-Applikation zu schaffen, wurden folgende Java-Klassen bereitgestellt. Füge diese in das entsprechende Package unter src/main/java in dein Projekt ein.

- `de.uniks.pmws2223.Main` [2]
- `de.uniks.pmws2223.App` [3]
- `de.uniks.pmws2223.controller.Controller` [4]
- `de.uniks.pmws2223.controller.LoginController` [5]
- `de.uniks.pmws2223.controller.BattleController` [6]

2.3 Funktionalität

In dieser Aufgabe soll eine Verbindung zwischen fxml-Dateien und Controller geschaffen werden. Implementiere hierzu die mit `// TODO` markierten Methoden der kopierten Klassen. Durch Kommentare in den Methoden wird deren Funktionalität bereits vorgegeben. Nach Bearbeitung der Aufgabe sollte es möglich sein, durch das Klicken des Start-Buttons vom Login-Bildschirm in den Battle-Bildschirm zu gelangen. Ebenso sollte dies umgekehrt durch den Leave-Button möglich sein.

Weiterhin sollen an dieser Stelle die **Fenstertitel** umgesetzt werden, die in den Wireframes der vorherigen Aufgabe zu sehen sind. In Abbildung 1 ist dieser „PMon - Main Menu“ und in Abbildung 2 „PMon - Encounter“.

Sollten die Fenstertitel deiner eigenen Wireframes abweichen, setze trotzdem die hier genannten Fenstertitel um. Du kannst die Fenstertitel deiner eigenen Wireframes gerne an die hier genannten anpassen.

Committe und pushe die Änderung abschließend auf den main-Branch.

Achte darauf, das Repository der aktuellen Hausaufgabe zu verwenden.

[1] https://github.com/sekassel/pmws2223-files/blob/main/HA06/build.gradle
[2] https://github.com/sekassel/pmws2223-files/blob/main/HA06/Main.java
[3] https://github.com/sekassel/pmws2223-files/blob/main/HA06/App.java
[4] https://github.com/sekassel/pmws2223-files/blob/main/HA06/Controller.java
[5] https://github.com/sekassel/pmws2223-files/blob/main/HA06/LoginController.java
[6] https://github.com/sekassel/pmws2223-files/blob/main/HA06/BattleController.java

Aufgabe 3 – TestFX (7P)

Als Abschluss für diese Hausaufgabe sollen die eben implementierten Teile getestet werden. Dies geschieht wie in der Vorlesung gezeigt mithilfe der hinzugefügten TestFx-Library.

3.1 Vorgegebene Klasse

Nutze die zur Verfügung gestellte Test-Klasse AppTest[7], um mit der Implementierung des TestFx-Tests zu beginnen. Füge den AppTest unter src/test/java in das Package de.uniks.pmws2223 in dein Projekt ein.

3.2 Implementierung

Der changeView-Test soll folgenden Ablauf implementieren:

- Initialen Fenstertitel prüfen
- Spielernamen „Alice“ in das dafür vorgesehenen Eingabefeld eingeben.
- Start-Button klicken, um ein Encounter zu starten
- Neuen Fenstertitel prüfen
- Leave-Button klicken
- Fenstertitel erneut prüfen
- Prüfen, dass das Eingabefeld für den Spielernamen leer ist

Committe und pushe die Änderung abschließend auf den main-Branch.

Achte darauf, das Repository der aktuellen Hausaufgabe zu verwenden.

[7] https://github.com/sekassel/pmws2223-files/blob/main/HA06/AppTest.java

Anhang

Es folgt eine Auflistung hilfreicher Webseiten und weiterer Erklärungen zu den Themen dieser Hausaufgabe. Die Links sind als Startpunkt zur selbstständigen Recherche angedacht. Das Durcharbeiten der folgenden Quellen ist kein bewerteter Anteil der Hausaufgaben.

Zur Verfügung gestellte Dateien

- https://github.com/sekassel/pmws2223-files/tree/main/HA06

Scene Builder

- Download: https://gluonhq.com/products/scene-builder/

TestFX

- Beispiele auf GitHub: https://github.com/testfx/testfx#examples