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 19.01.2023 - 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!

Aufgabe 1. Speichern und Laden (18P)

PMon soll nun durch ein System für Speichern und Laden von sogenannten Save-Slots erweitert werden. Du kannst dich dabei an folgendem Wireframe orientieren:

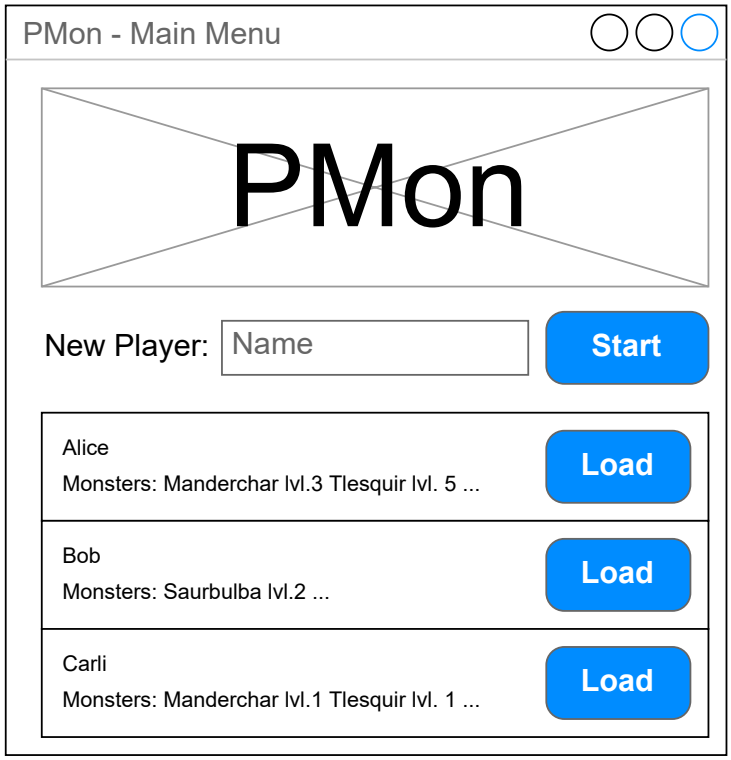

Abbildung 1: Login mit Save-Slots

1.1 Gradle

Füge in der Datei build.gradle unter dependencies folgende Zeilen hinzu:

```
// https://mvnrepository.com/artifact/org.fulib/fulibYaml
implementation group: 'org.fulib', name: 'fulibYaml', version: '1.5.0'
```

1.2 PlayerRepository

Lege folgende Methoden zum Speichern und Laden von Spielern in der neuen Klasse PlayerRepository im Package de.uniks.pmws2223.service im Modul src/main/java an:

```
public Player loadPlayer(String name)
public List<Player> loadPlayers()
public void savePlayer(Player player)
```

Implementiere die Methoden analog zu Vorlesung und Übung, sodass sie Spieler unter dem Pfad data/<Player Name>.yaml im YAML-Format speichern und laden.

Hinweis: Spielernamen mit Sonderzeichen wie `\/:*?"<>|` können zu Problemen führen. Es ist erlaubt, diese Zeichen in den Dateinamen zu ersetzen, beispielsweise durch Unterstriche mit replace.

1.3 Laden

Erweitere den Login-Bildschirm um eine Liste von Save-Slots. Diese können anhand des Spielernamens identifiziert und dargestellt werden. Jeder Save-Slot soll einen Button zum Laden anbieten. Dieser wechselt beim Klick mit dem zugehörigen Spieler in den Battle-Bildschirm und startet ein neues Encounter.

1.4 Speichern

Die bereits existierende Eingabemöglichkeit für Spieler wird nun verwendet, um einen neuen Save-Slot anzulegen. Zusätzlich zum Wechsel in den Battle-Bildschirm soll nun auch der neu erstellte Spieler gespeichert werden.

Erweitere den Battle-Bildschirm, sodass er bei einem Sieg oder einer Niederlage den Spieler samt Monstern mithilfe des PlayerRepository abspeichert. Achte darauf, das encounter des Spielers zu entfernen, damit es nicht gespeichert wird.

Aufgabe 2. Spiel fertigstellen (11P)

In dieser Aufgabe musst du die letzten Funktionalitäten für das Spiel implementieren.

2.1 Game Over

Erstelle einen neuen GameOver-Bildschirm mit einem GameOverController. Dieser soll geöffnet werden, wenn der Spieler ein Encounter gewinnt oder verliert. Das Ergebnis des Kampfes soll entsprechend dargestellt werden (z.B. „Du hast gewonnen!“).

Füge einen Button hinzu, der den Benutzer zurück zum Hauptmenü (Login) bringt, sowie einen Button, der ein neues Encounter mit dem gleichen Spieler erstellt und im Battle-Bildschirm startet.

2.2 Monster wechseln

Erweitere den MonsterController und den BattleService, sodass der Benutzer ein Monster in den Kampf schicken und damit das vorherige Monster auswechseln kann. Füge dazu im MonsterController eine Aktion ein, beispielsweise einen Button oder einen Click-Handler auf der umliegenden Box.

Die neue Methode selectMonster im BattleService soll die Logik dieser Funktionalität bereitstellen, indem er prüft, ob das ausgewählte Monster kampffähig (>0HP) ist, die Änderung im Datenmodell durchführt, und eine Ausgabe im Battle Log erstellt.

Schreibe einen selectMonster-Test im BattleServiceTest mit den zwei möglichen Fällen (kampffähiges/kampfunfähiges Monster ausgewählt).

Aufgabe 3. Final Testination (9P)

Zu guter Letzt muss der kritische Pfad der Anwendung getestet werden. Das bedeutet, dass ein Test die nötigen Aktionen durchführen muss, damit ein Spieler das Spiel gewinnt.

Erweitere dafür den AppTest mit folgenden Aktionen nach dem Ablauf aus Hausaufgabe 7 (d.h. nachdem der Spieler gewonnen hat; der Game-Over-Bildschirm sollte nun sichtbar sein):

- Fenstertitel prüfen (PMon - Game Over)
- Nächstes Encounter starten
- Fenstertitel prüfen (PMon - Battle)
- Encounter sofort verlassen
- Fenstertitel prüfen (PMon - Main Menu)
- Prüfen, ob der Spieler nun im Hauptmenü erscheint
- Spieler laden
- Fenstertitel prüfen (PMon - Battle)
- Prüfen, ob die neuen Werte der Monster korrekt geladen wurden

Alle Tests (alt/neu) müssen nach wie vor funktionieren, sollte dies nicht der Fall sein, wird diese Aufgabe mit 0 Punkten bewertet!

Abschluss

Nach Implementierung der Funktionalität dieser Hausaufgabe ist die Anwendung PMon für dieses Semester **abgeschlossen**. Weitere Features dürfen bei gegebener intrinsischer Motivation umgesetzt werden, sind jedoch nicht Pflicht. In diesem Fall sollte jedoch der Commit, welcher die Aufgabenstellung abschließt, deutlich markiert werden, damit eventuelle Erweiterungen nicht zu Abzügen führen.

In der kommenden Hausaufgabe für Studiengänge mit 6CP wird eine neue Anwendung begonnen.

Für die Studierenden der Mechatronik (4CP) ist diese (Hausaufgabe 8) die letzte reguläre Hausaufgabe. In der kommenden Woche wird das Aufgabenblatt zum Projekt für Studierende der Mechatronik veröffentlicht.