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/ws2021/programming-and-modelling/` zu berücksichtigen.

Abgabefrist ist der 28.01.2021 - 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 benötigt ihr **das bestehende MiniRPG-Repository**. Falls noch nicht geschehen, kann es über folgenden Link angelegt werden:

`https://classroom.github.com/a/5LkDnrR7`

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

Vorbereitungen

Anpassungen der build.gradle

Um das Speichern und Laden, welches in dieser Hausaufgabe Thema ist anhand von Vorlesung und Übung zu implementieren, muss eine Anpassung an der build.gradle vorgenommen werden. Um Copy-Paste Fehlern vorzubeugen findet ihr die hinzuzufügenden Zeilen auf unserem Blog im dem .zip-Archiv. In diesem Archiv befinden sich noch weitere Dateien, welche für diese Hausaufgabe genutzt werden, mehr hierzu in den entsprechenden Aufgaben.

Anschließend sollte ein Gradle Refresh durchgeführt werden.

Aufgabe 1 - Property Change Listener (24P)

Ziel dieser Aufgabe ist es, Änderungen des Datenmodells während der Laufzeit in der Oberfläche anzuzeigen. Dafür werden PropertyChangeListener verwendet, welche an Modellobjekten angemeldet werden können.

1.1 Button Funktionalität

Verknüpft zuerst die folgenden Buttons des DungeonScreens (vgl. HA7 Abbildung 1) mit den entsprechenden Methoden des RPGEditors:

- Upgrade Stat (HeroStatView) -> heroStatUpdate(HeroStat stat)
- Attack/Defense (HeroView) -> heroEngagesFight(String heroStance, Hero hero)

Am Ende dieser Aufgabe müssen alle Buttons eurer MiniRPG-Anwendung mit der entsprechenden Funktionalität versehen sein.

1.2 PropertyChangeListener

Für folgende Änderungen sollen PropertyChangeListener am Datenmodell angemeldet werden:

- Anzahl besiegter Enemies
- Hero-LP
- Hero-Coins
- Aktuell bekämpfter Enemy, inkl. dessen Name
- Enemy-LP
- Enemy-Pose
- Stat-Level
- Stat-Value
- Stat-Cost

Wichtige Hinweise zur Implementierung:

- PropertyChangeListener werden in init() angemeldet.
- Damit PropertyChangeListener wieder abgemeldet werden können, müssen sie in Feldern gespeichert werden.
- PropertyChangeListener müssen korrekt abgebaut werden, wenn die stop()-Methode aufgerufen wird.
- Eventuell müssen PropertyChangeListener nicht nur in init() und stop(), sondern auch bei Änderung von Objekten an- und abgemeldet werden.

Aufgabe 2 - Speichern & Laden (19P)

Ziel dieser Aufgabe ist es, einzelne Heros speichern und eine Liste von Heroes laden zu können. Kopiert hierzu den ResourceManager aus den Vorlagen und fügt ihn in das Package de.uniks.pmws2021.util ein. Implementiert die Methoden saveHero() und loadAllHeroes() anhand der Kommentare in der Vorlage.

Speichern

Ein Hero soll automatisch nach Besiegen des letzten Enemies, also dem Abschließen eines Dungeons, gespeichert werden. Ruft hierzu an einer geeigneten Stelle in eurer Applikation die saveHero()-Methode des ResourceManagers auf.

Laden

Beim Öffnen des HeroScreens (vgl. HA6 Abbildung 1) sollen alle zuvor abgespeicherten Heroes automatisch geladen und angezeigt werden. Ruft hierzu an einer geeigneten Stelle in eurer Applikation die loadAllHeroes()-Methode des ResourceManagers auf. Es muss möglich sein, einen Dungeon mit einem zuvor geladenen Hero zu betreten.

Hinweis:

Für das Speichern und Laden dürfen keine zusätzlichen Buttons zur Applikation hinzugefügt werden.
Die erstellte Datei darf nicht mit Git gepusht werden. Trage dafür den Ordner, in dem die Daten gespeichert werden, in deine .gitignore ein.

Aufgabe 3 - More Testing (17P)

In dieser Aufgabe wird die Funktionalität aus den ersten beiden Aufgaben getestet. Füge die folgenden Klassen aus den Vorlagen in dein Projekt ein.

`de.uniks.pmws2021.PropertyChangeTest`
`de.uniks.pmws2021.util.SaveLoadTest`

3.1 PropertyChangeTest

In diesem Aufgabenteil sollen **alle Änderungen** (vgl. Aufgabe 1.2) am Datenmodell und dessen Visualisierung getestet werden. Geht im PropertyChangeTest folgende Schritte ab:

1. DungeonScreen öffnen
2. Prüfen, ob die View die initialen Daten anzeigt
3. Daten im Modell ändern, z.B. hero.setLp(...), hero.setAttacking(...), stat.setLevel(...) (Hierdurch werden die PropertyChangeEvents erzeugt)
4. Prüfen, ob die View die geänderten Daten anzeigt

3.2 SaveLoadTest

In diesem Aufgabenteil soll das Speichern und Laden aus Aufgabe 2 getestet werden. Geht in der Test-Methode folgende Schritte ab:

- Alte Speicherdatei löschen, falls vorhanden
- Zwei Hero Objekte mit beliebigen Werten erstellen
- Beide Heroes mittels ResourceManager speichern
- Prüfen, ob die Heroes in der Datei gespeichert wurden
- Liste der Heroes über den ResourceManager laden
- Prüfen, ob zwei Hero-Objekte geladen wurden und die zuvor gesetzten Werte haben

Abschluss:

Nach Implementierung der Funktionalität dieser Hausaufgabe ist die Anwendung MiniRPG für dieses Semester abgeschlossen. Einige wenige fehlende Features dürfen bei gegebener intrinsischer Motivation umgesetzt werde, sind jedoch nicht Pflicht. In der kommenden Hausaufgabe wird eine neue Anwendung begonnen.