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 21.01.2021 - 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.

Aufgabe 1 - RPGEditor (9P)

In dieser Aufgabe soll der RPGEditor in das bereits bestehende Projekt eingepflegt werden. Hierfür müssen folgende Schritte ausgeführt werden, um bereits zuvor geschriebene Funktionalität zu übernehmen.

Aufgabe 1.1 Editor einpflegen

Kopiert den RPGEditor aus den Vorlagen in euer Projekt.

Aufgabe 1.2 Funktionalität übertragen

Wie der Name des TemporaryGameController bereits vermuten ließ, war dessen Verwendung nur temporär. Folgende Methoden müssen aus dem TemporaryGameController in die Methoden im RPGEditor überführt werden:

```
enterDungeon(Hero hero)
heroStatUpdate(HeroStat stat)
heroEngagesFight(String heroStance, Hero hero)
evaluateFight(Enemy enemy, Hero hero)
```

Aufgabe 1.3 Weitere Methoden implementieren

Implementiert die verbleibenden have-Methoden im Editor. Die Implementierung kann dabei individuell angepasst werden (zB. Random Generated Enemies).
Wichtig ist, dass am Ende dieser Aufgabe neue Objektinstanzen nur noch in den have-Methoden angelegt werden.

Aufgabe 2 - Modulare View (17P)

In dieser Aufgabe soll die einheitliche Oberfläche aus Hausaufgabe 06 in modular nutzbare Oberflächenelemente aufgeteilt werden.
Als Faustformel gilt: Jedes dargestellte Objekt aus dem Datenmodel wird von einem eigenen Controller verwaltet. Jeder Controller steuert dabei eines der modularen Oberflächenelemente.

Abbildung 1 zeigt den jeweiligen Handlungsbereich der Controller und zugleich die Aufteilung der Oberfläche in ihre Oberflächenelemente.

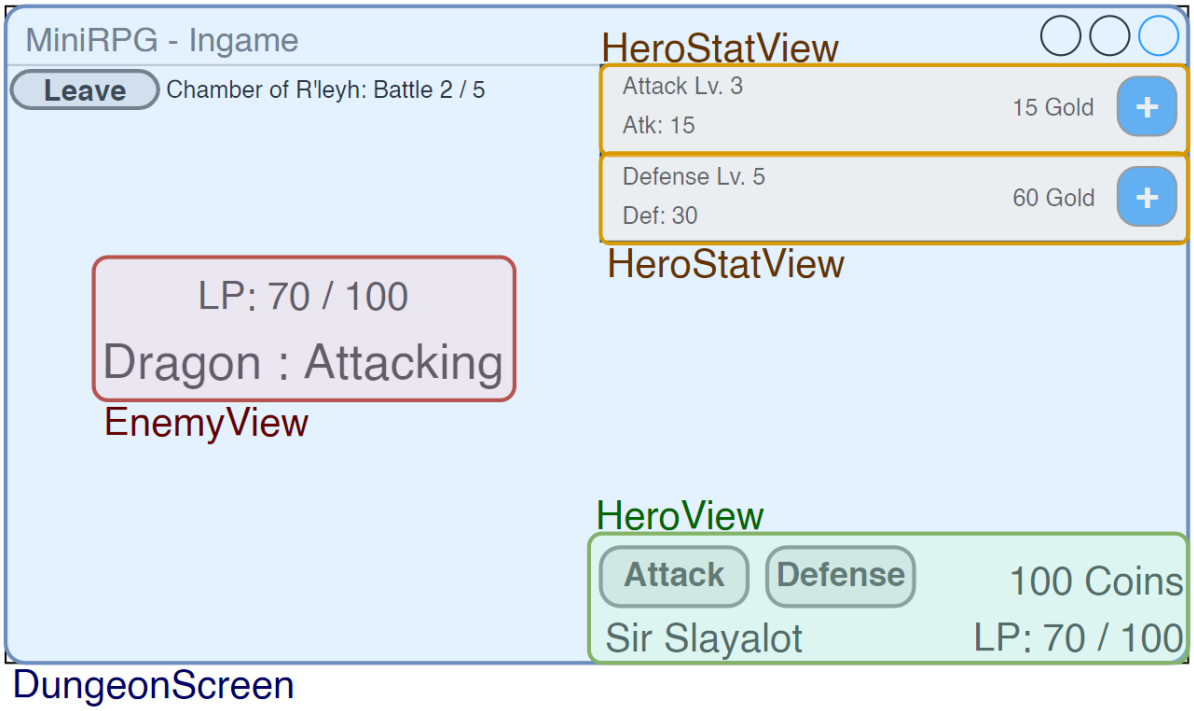

Abbildung 1: Beispielaufteilung (Sub)-Controller

Fügt die vorgegebenen SubControllerklassen aus den Vorlagen in euer Projekt ein. (Achtet dabei auf den subcontroller Zusatz im Package).
Kopiert auch den Zusatz für die DungeonScreenController in dessen init()-Methode.
Unterteilt euren DungeonScreen im Anschluss, sodass jeder Subcontroller die Verwaltung des entsprechenden Subviews gewährleistet.

Diese FXMLs sollen HeroView.fxml, EnemyView.fxml und HeroStatView.fxml heißen.

Hinweis: Enemy- und HeroSubView können in den DungeonScreen included werden, während die HeroStatSubViews dynamisch nachgeladen werden muss.

Aufgabe 3 - Daten-Verknüpfung (49P)

In dieser Aufgabe sollen die Daten aus dem Datenmodell in der View angezeigt werden.

Aufgabe 3.1 View mit Daten verknüpfen

Hierzu sollen die Viewelemente, z.B. Labels, mit den Daten versehen werden. Die Daten entstammen dem model-Objekt eines Controllers. Die Zuweisung der Daten ist in der init-Methode des jeweiligen Controllers umzusetzen.

Des Weiteren sollen die Eingaben aus dem HeroScreen verwendet werden, um einen Dungeon zu initialisieren. Hierzu muss nach Betätigung des Create&Start-Buttons die Information zu Namen und Spielmodus ausgelesen werden. Diese werden dann der enterDungeon-Methode übergeben. Erst danach erfolgt der Übergang zum DungeonScreen.

Aufgabe 3.2 Verbindung testen

Passt die changeViewTest-Methode so an, dass dieser den zuvor eingegeben Hero Namen im DungeonScreen prüft.

Zusätzlich soll eine weitere Testmethode oder Testklasse entstehen, welche nach Initialisierung des DungeonScreens überprüft, ob die Daten aus dem Datenmodell korrekt im View angezeigt werden:

- Name des Dungeons
- Anzahl an Enemies
- LP des aktuellen Enemys
- Stance des aktuellen Enemys
- LP des Heros
- Coins des Heros

Aufgabe 3.3 Neujahrsputz

Im Anschluss muss die Klasse TemporaryGameController gelöscht werden, da diese durch den RPGEditor ersetzt wurde. Achtet nach dem Löschen der Datei darauf, dass euer Projekt weiterhin kompiliert und somit alle Referenzen zum TemporaryGameController entfernt/ersetzt wurden.