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 03.12.2020 - 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 ein **neues Repository**, das über folgenden Link angelegt wird:

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

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

## Vorbereitung

Zur Bearbeitung der Hausaufgabe sollte eine Entwicklungsumgebung verwendet werden. Wir empfehlen aufgrund der Nachvollziehbarkeit die Verwendung von IntelliJ (siehe Aufgabenblatt 3). Die Abgabe muss als **lauffähiges** Projekt abgegeben werden.

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

### Zukünftige Abgaben

Das oben genannte Repository ist der Startpunkt für die Anwendung, die im weiteren Verlauf dieser Veranstaltung erstellt werden soll. Das Repository wird also fortan nicht für jede Hausaufgabe gewechselt, sondern für diverse kommende Abgaben weiterverwendet.

## Aufgabe 1 - Fulib.org (50P)

In dieser Aufgabe nutzen wir die Funktionalität der Fulib.org-Plattform. Diese ermöglicht das einfache Erstellen eines Gradle-Projekts zusammen mit einem Datenmodell. Es stellt somit ein Tool dar, das alle Schritte, die ein herkömmliches Projektsetup (wie in Hausuafgabe 3) mit sich zieht, deutlich einfacher gestaltet.

`https://www.fulib.org/`

Arbeitet die auswählbaren Beispiele der Webseite durch, um euch mit der Syntax und Semantik dieser speziellen Form der Szenarios vertraut zu machen. Beachtet dabei, dass die in Hausaufgabe 1 bis 3 vorgestellten Szenarios sogenannte Freitext-Szenarien darstellen. Die Szenarien der Webseite erfordern keine Unterteilung in Title, Start, Action und End.

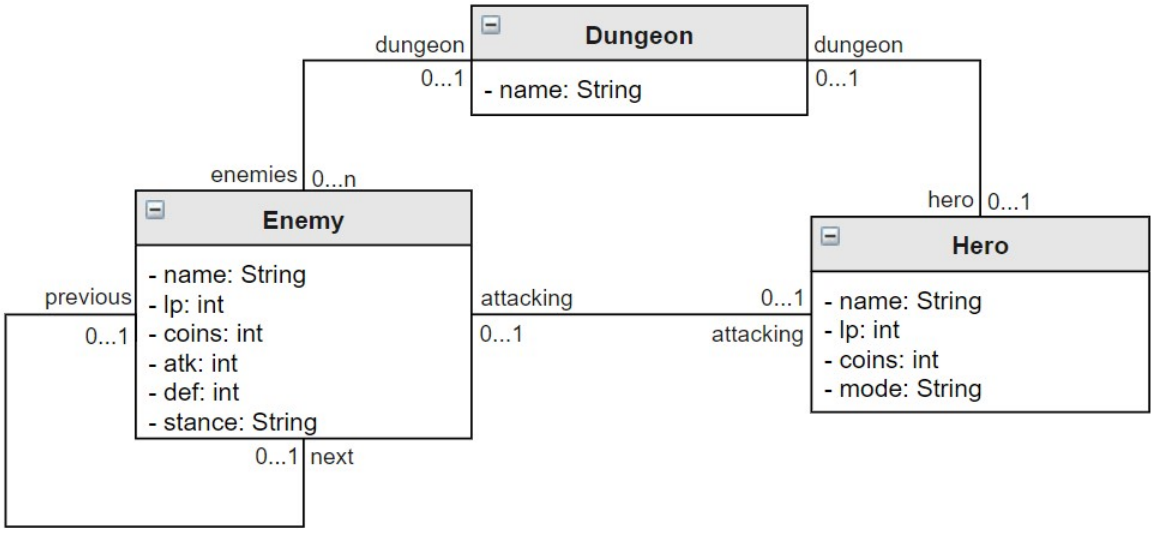


Abbildung 1: Ausschnitt aus Klassendiagramm

1. Erstelle nun ein Szenario für das Klassendiagramm in Abbildung 1. Das abgebildete Klassendiagramm und jenes, welches im „Class Diagram"-Teil der Weboberfläche angezeigt werden, sollten sich gleichen, bevor der Folgeschritt durchgeführt wird.

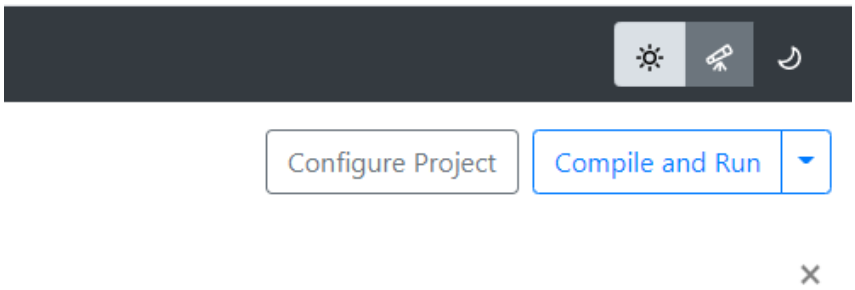


Abbildung 2: Projekt-Button auf Fulib.org

2. Lade nun über den in Abbildung 2 gezeigten „Configure Project"-Button das erstellte Klassendiagramm als lauffähiges Projekt herunter. In dem sich öffnenden Fenster tragt ihr dann folgende Dinge ein:

| | | |
|---|---|---|
| Package/Group Name | : | de.uniks.pmws2021.model |
| Project Name | : | PMWS2021_MiniRPG_<GitHubName> |
| Version | : | 0.1.0 |
| Scenario File Name | : | Scenario.md |
| Decorator Class Name | : | GenModel |

   Anschließend kann das Projekt mit dem grünen „Download"-Button als Gradle-Projekt in einer .zip Datei herunterladen werden.

3. Tausche die im Projekt enthaltene Gitignore durch die aus Hausaufgabe 3 bekannte Datei aus, oder ersetze dessen Inhalt manuell. Committe erst danach die Dateien des Projektes.

4. Führe den Gradle Task check aus, um das im Szenario beschriebene Model zu generieren. Der Task kann in IntelliJ im Gradle-Seitenmenü unter Tasks/verification gefunden werden.

Committe und pushe die Änderung an deinem Gradle-Projekt abschließend auf den main-Branch.

**Bei der Bewertung wird vor allem auf die korrekten Kardinalitäten zwischen den Klassen geachtet.**

**Achtet darauf, das Repository der aktuellen Hausaufgabe zu verwenden.**

## Aufgabe 2 - Fulib (25P)

In dieser Aufgabe wird das vollständige Datenmodell des Spiels "MiniRPG" mit der Hilfe von Fulib generiert. Wie der Name bereits vermuten lässt, ist Fulib die treibende Technologie hinter Fulib.org. Diese kann in bestehenden Projekten genutzt werden, um noch präzisere Definitionen für das Datenmodell zu tätigen.

Wir nutzen die bereits generierte Klasse GenModel, um das im Scenario definierte Klassenmodell um die Klassen HeroStat, AttackStat und DefenseStat zu erweitern. Ziel ist, alle in Abbildung 3 gezeigten Klassen zu generieren.

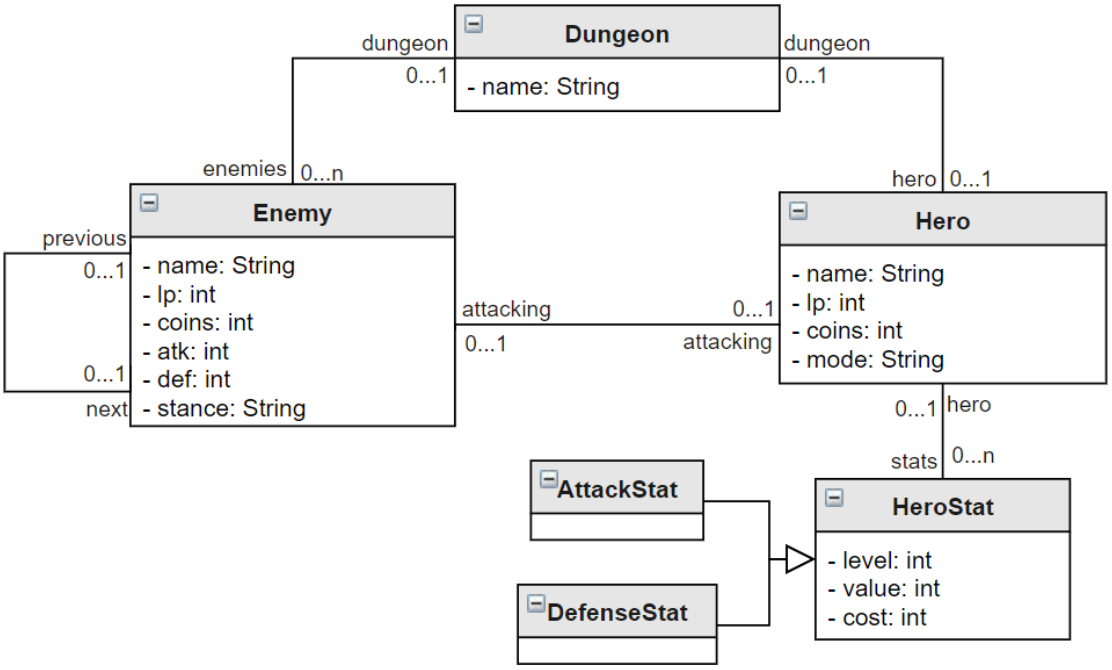


Abbildung 3: „MiniRPG"-Klassendiagramm

**Hinweis:**

Die Kanten mit weißen Pfeilköpfen und ohne Bezeichnungen oder Kardinalitäten beschreiben die Vererbung. So erben AttackStat und DefenseStat beispielsweise von der Klasse HeroStat.

Für die Variable stance werden in einer zukünftigen Aufgabe Konstanten definiert. Für Hausaufgabe 04 definieren wir stance vorläufig als String-Feld.

Committe und pushe die Änderung an deinem Gradle-Projekt abschließend auf den main-Branch.

**Bei der Bewertung wird vor allem auf die Erweiterung des Modells und die Vererbung von Klassen geachtet.**

**Achtet darauf, das Repository der aktuellen Hausaufgabe zu verwenden.**

## Aufgabe 3 - Initialisierung (25P)

In dieser Aufgabe soll die enterDungeon()-Methode aus **Hausaufgabe 03** an das zuvor generierte Datenmodell angepasst werden:

- Erstelle eine Klasse TemporaryGameController.java unter src/main/java im Package de.uniks.pmws2021.controller. Diese ist der Ausgangspunkt unserer Logik.
- Erstelle nun die Methode enterDungeon(Hero hero) im TemporaryGameController. Die Methode muss am Ende ein Dungeon-Objekt zurückgeben. Nur innerhalb dieser Methode dürfen Instanzen einer Klasse (mit Ausnahme von Hero) initialisiert werden.
- Das initiale Spielgeschehen muss ein Dungeon mit 3 Enemies enthalten, welche wiederum in einer festen Reihenfolge verbunden sind. Weiterhin muss ein Hero übergeben werden, welcher mit dem ersten Enemy des Dungeons verbunden ist. Bitte achte darauf, dass einige Änderungen am Datenmodell und Kardinalitäten (previous - next) vorgenommen wurden.
- Die weiteren Werte der Objekte lauten:
  Hero: Name "Sir Slayalot", LP 100, coins 30, ATK (lv 2, value 15, basecost 5), DEF (lv 3, value 20, basecost 5), normal mode
  Enemy 1: Name "Goblin" with 30 LP, 5 ATK, 7 DEF and 5 coins
  Enemy 2: Name "Ogre" with 60 LP, 10 ATK, 10 DEF and 7 coins
  Enemy 3: Name "Hydra" with 50 LP, 20 ATK, 8 DEF and 10 coins
- Alle Objekte müssen mit dem „Dungeon"-Spiel verbunden sein.
- Der TemporaryGameController muss über eine zu erstellende Testklasse ausführbar sein, die die enterDungeon()-Methode mit einem Hero als übergebenem Parameter aufruft und im Nachhinein ein Objektdiagramm mit FulibTools ausgibt.
- Für das Erstellen eines Hero-Objektes kann nach eigenem Ermessen eine Methode in der Klasse TemporaryGameController implementiert werden.

Committe und pushe die Änderung an deinem Gradle-Projekt abschließend auf den main-Branch.

**Bei der Bewertung wird vor allem auf die vollständige Umsetzung der Spielsituation geachtet.**

**Achtet darauf, das Repository der aktuellen Hausaufgabe zu verwenden.**