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 26.11.2020 - 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/62qqh4HD`

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 Anhang). Die Abgabe muss als **lauffähiges** Projekt abgegeben werden.

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

Java

Wir verwenden für die Veranstaltung Java 11. Dieses könnt ihr unter folgendem Link herunterladen und installieren:

`https://adoptopenjdk.net/releases.html?variant=openjdk11&jvmVariant=hotspot`

Wählt dort die für euer Betriebssystem passende Datei aus und installiert diese. Ihr seid weiterhin frei, andere Java-Versionen oder Anbieter für JDKs bzw. JREs zu nutzen. Da die korrekte Konfiguration bei selbst gewählten Anbietern von eurem Betriebssystem, dessen Version und anderen Parametern abhängt, seid ihr hier jedoch auf die eigenen Fähigkeiten angewiesen. Wir können bei der Wahl dieser Methode keine Hilfestellung geben.

Gradle-Projekt aufsetzen

Erstellt ein Gradle-Projekt wie in der Übung gezeigt. Dieses soll den Namen PMWS2021_Assignment03_<GitHubName> tragen.
Noch vor dem ersten Commit müsst ihr folgende Datei in das Projekt hinzufügen:

Gitignore

Füge eine Gitignore zu deinem Projekt hinzu. Diese muss im Root-Verzeichnis des Projektes liegen (also parallel zu dem .git-Verzeichnis). Ihr findet diese Gitignore zum Herunterladen auf unserem Blog neben dem Link zu dieser Hausaufgabe.

Das Projekt sollte danach ungefähr solch eine Struktur aufweisen.

```
PMWS2021_Assignment03_<GitHub-Name>/
├── .git/...
├── src/...
├── .gitignore
├── build.gradle
└── ...
```

Erst nach erfolgreichem Hinzufügen der Gitignore sollt ihr den ersten Commit machen und das Projekt in das Repository pushen. Dieser Schritt muss vorgenommen werden, da ansonsten Projektdateien hochgeladen werden, die so spezifisch für euer System sind, dass sie das Herunterladen für uns als Korrekteure und somit auch das Korrigieren unnötig erschweren.

Ein Nicht-Einbinden der Gitignore führt zu Punktabzug.

build.gradle

Füge die Abhängigkeit zu FulibTools in die build.gradle Datei wie unten gezeigt hinzu.

```
repositories {
    mavenCentral()
    jcenter()
}

dependencies {
    ...
    testCompile group: 'org.fulib', name: 'fulibTools', version: '1.4.0'
}
```

WICHTIG: Das Kopieren der Zeilen aus der PDF führt zu einer fehlerhaften Formatierung des Textes. Tippt diese per Hand!

Aufgabe 1 - Klassendiagram to Java (60P)

In dieser Aufgabe implementieren wir ein vereinfachtes Klassendiagramm des Spiels "MiniRPG". Hierzu ist das in Abbildung 1 dargestellte Klassendiagramm gegeben.

Wie bereits in der Vorlesung angesprochen, bitten wir euch, während der Bearbeitung dieser Aufgabe 1, eure Bearbeitungszeit zu erfassen. Legt hierfür eine Datei mit dem Namen time.txt in euer Projekt (gerne parallel zur .gitignore). Es reicht aus, wenn am Ende die Summe der Stunden/Minuten eingetragen wurde, die für die Bearbeitung von Aufgabe 1 aufgebracht wurde.

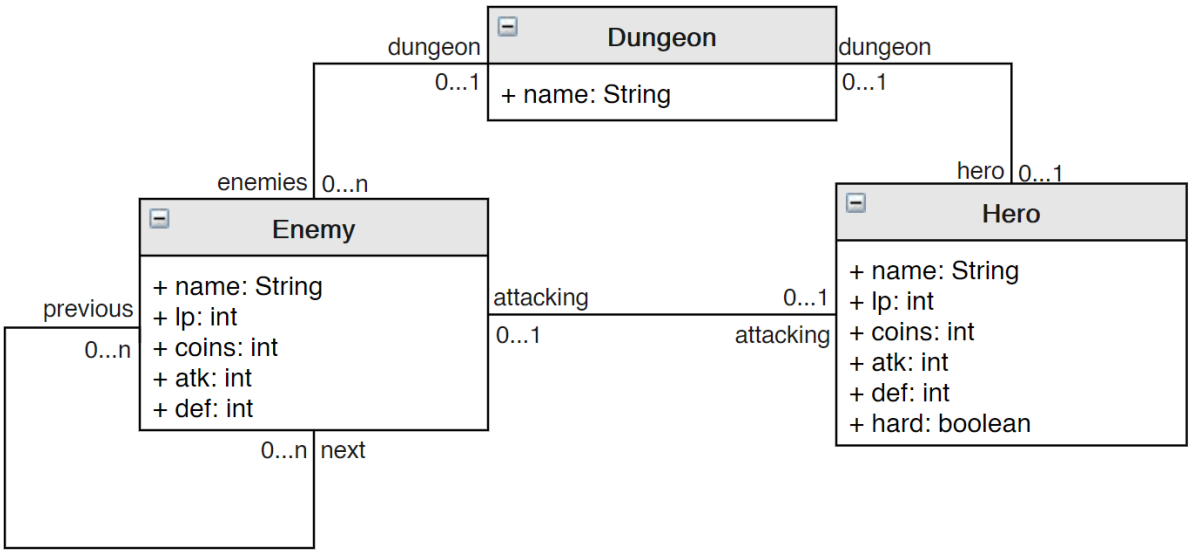

Abbildung 1: MiniRPG-Klassendiagram (vereinfacht)

Folgende Vorgehensweise wird vorgeschlagen:

1. Erstelle für jede Klasse im Diagramm eine Klasse in einer separaten .java-Datei unter dem Modul src/main/java im zu erstellenden Package de.uniks.ws2021.minirpg.model
2. Füge den Klassen keine Konstruktoren hinzu.
3. Füge den Klassen die entsprechenden Attribute hinzu. Achte dabei auf die Zugriffsverkapselung der Attribute mit den Zugriffsmethoden Set und Get aus der Vorlesung!
   - <Class> set<Field>(<Type> <Variable>) / <Type> get<Field>() /
4. Implementiere die Assoziationen und stelle die referenzielle Integrität sicher. Dies verlangt folgende Punkte:
   - Füge den Klassen für 0...1-Assoziationen die Zugriffsmethoden <Type> get<Field>() / <Class> set<Field>(<Class> <Variable>) hinzu
   - Füge den Klassen für 0...n-Assoziationen die Zugriffsmethoden List<Type> get<Field>() / <Class> with<Field>(<Class> <Variable>) / <Class> without<Field>(<Class> <Variable>) hinzu
   - Wähle für die 0...n-Assoziationen eine geeignete Containerklasse (z.B. ArrayList)
   - Sorge dafür, dass bei allen bidirektionalen Assoziationen die Rückrichtung automatisch mitgesetzt wird.

Beispielsweise, dass beim Setzen eines neuen Dungeon für einen Enemy als dessen Dungeon, gleichzeitig auch der gewählte Dungeon den Enemy in seine Collection an Enemies aufnimmt und (**wichtig!**) der Enemy aus der Collection seines vorherigen Dungeon entfernt wird.

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

Bei der Bewertung wird vor allem auf die Projektstruktur, Zugriffsverkapselung sowie die korrekte Umsetzung der referenziellen Integrität geachtet.

Achtet darauf, das Repository der aktuellen Hausaufgabe zu verwenden.

Aufgabe 2 - Testing (40P)

In dieser Aufgabe soll anhand eines Szenarios (Abbildung 2) und dessen Objektdiagrammen (Abbildung 3 und 4) ein Test für die Implementierung von Aufgabe 1 entstehen. Hierzu ist folgendes Szenario gegeben:

Title: "Sir Slayalot" enters a new Dungeon

Start:	Player Bob is in the hero selection screen. There are two Heros he can choose. "Sir Slayalot" (LP 100, coins 30, ATK 2, DEF 3, normal mode) and "Lady Slayna" (LP 100, coins 30, ATK 5, DEF 5, hard mode).
Action:	Bob chooses the Hero "Sir Slayalot" to enter a new dungeon.
Result:	"Sir Slayalot" enters the Dungeon "Bottom of the Pit". The Dungeon is guarded by 3 monsters. First is a "Goblin" with 30 LP, 5 ATK, 7 DEF and 5 coins. Second is an "Ogre" with 60 LP, 10 ATK, 10 DEF and 7 coins. Last is a "Hydra" with 50 LP, 20 ATK, 8 DEF and 10 coins. "Sir Slayalot" targets the Goblin. The Goblin takes an attacking pose. The player can choose his current move.

Abbildung 2: Szenario zum Dungeoneintritt

Für die Implementierung wird folgende Vorgehensweise vorgeschlagen:

1. Erstellt eine Klasse DungeonTest.java unter dem Modul src/test/java im zu erstellenden Package de.uniks.ws2021.minirpg.model.test
2. Erstellt einen Test enterDungeonTest(), in dem ihr zunächst den Zustand aus der Startsituation des Szenarios in Abbildung 2 herstellt.
3. Testet nun, ob die Objekte wie im Objektdiagramm (siehe Abbildung 3) miteinander verbunden sind. Es müssen mindestens 5 sinnvolle Asserts existieren.
4. Gebt das Objektdiagramm als Bilddatei mit dem Namen start.svg im Ordner diagrams/ aus. (Tipp: dieses könnt ihr nun mit Abbildung 3 vergleichen).
5. Implementiert eine neue Methode enterDungeon(Hero hero), die wie in der Vorlesung beschrieben die Action des Szenarios ausführt, und ruft diese anschließend im Test auf.
6. Testet nun nach der Ausführung der Action, ob die Objekte wie im Objektdiagramm (siehe Abbildung 4) miteinander verbunden sind. Es müssen mindestens 5 sinnvolle Asserts existieren.
7. Gebt das Objektdiagramm als Bilddatei mit dem Namen end.svg im Ordner diagrams/ aus. (Tipp: dieses könnt ihr nun mit Abbildung 4 vergleichen)

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

Bei der Bewertung wird vor allem auf die korrekte Nutzung von JUnit sowie die korrekte Umsetzung der Objektdiagramme geachtet.

Achte darauf, das Repository der aktuellen Hausaufgabe zu verwenden.

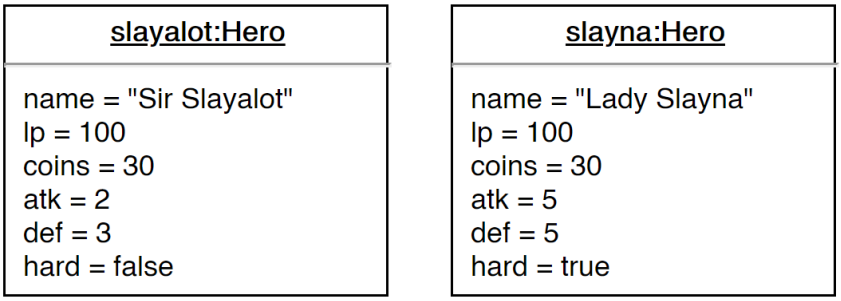

Abbildung 3: Objektdiagramm (Start)

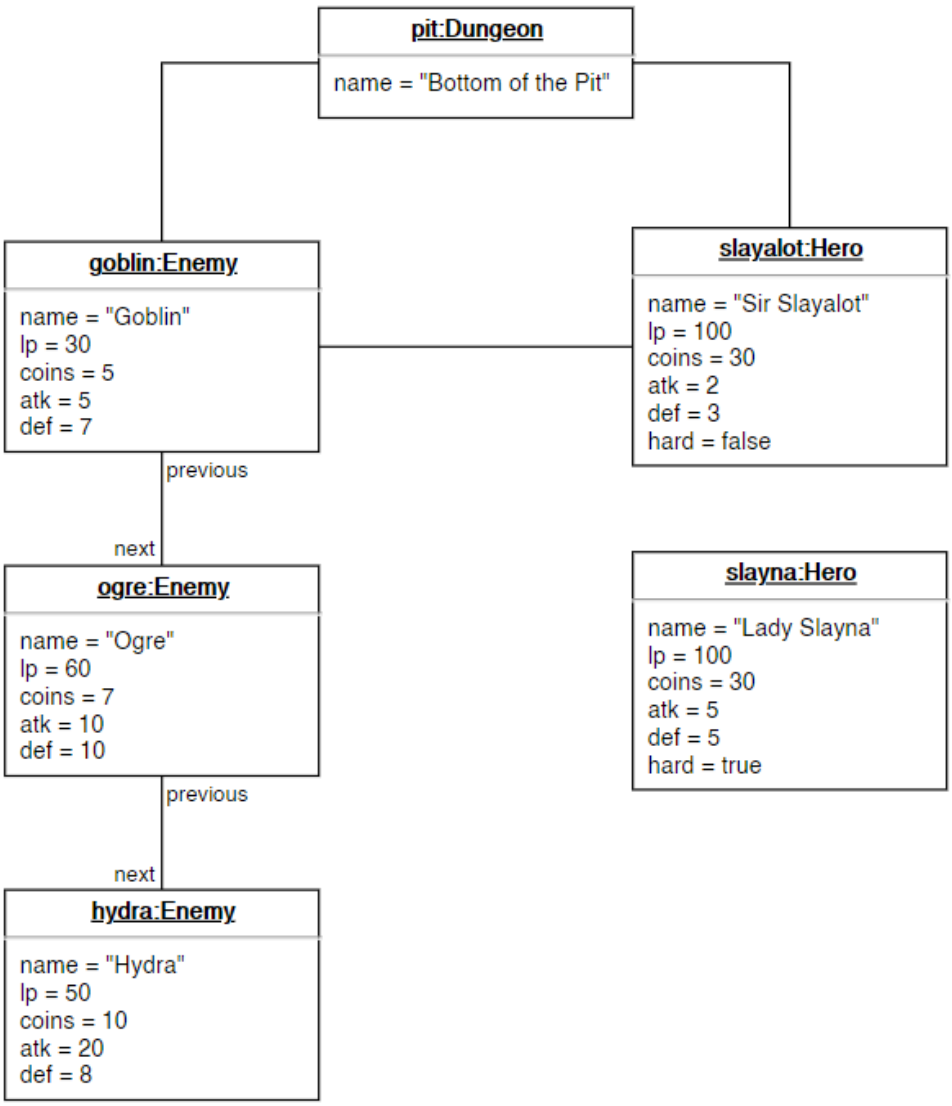

Abbildung 4: Objektdiagramm (Ende)

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.

GitKraken

- Download: `https://www.gitkraken.com/git-client`

Sourcetree

- Download: `https://www.sourcetreeapp.com/`
- Dokumentation:
  `https://confluence.atlassian.com/get-started-with-sourcetree`

FulibTools

- GitHub: `https://github.com/fujaba/fulibTools`

IntelliJ

- Download: `https://www.jetbrains.com/idea/download/`

  Die "Ultimate"-Version bekommt ihr kostenlos, nachdem ihr euch hier:
  `https://www.jetbrains.com/shop/eform/students` eine entsprechende Lizenz für Studierende geholt habt.
- Gradle-Projekt erstellen:
  `https://www.jetbrains.com/help/idea/getting-started-with-gradle.html`
- Projektstruktur: `https://www.jetbrains.org/intellij/sdk/docs/basics/project_structure.html`