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 10.12.2020 - 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 **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.
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!

Commit Message-Vorgaben

In dieser und zukünftigen Hausaufgaben führen wir eine Vorgabe für die Commit Messages ein.

Solche Commits, welche als Abgabe einer Hausaufgabe dienen, werden mit der Nachricht: ”Final HA<Number>” versehen. Für die Abgabe von Teilaufgaben (wie in Aufgabe 1 dieses Blattes), wird folgende Konvention eingeführt ”HA<Number> finished Task <TaskNumber>”.

Falls in Folge der Commits ein Fehler auffällt, welchen ihr zeitlich gesehen noch korrigieren könnt, so verwendet für den Folge-Commit wieder die Message ”Final HA<Number>” oder ”HA<Number> finished Task <TaskNumber>”.

Das Ignorieren der Vorgaben wird mit 0 Punkten bewertet!

Die Mockups müssen als PDF-Datei (.pdf), Bilddatei (.jpg, .png) oder Vektorgrafik (.svg) abgegeben werden. Jedes Mockup ist in einer eigenen Datei abzulegen.

Handschriftliche Abgaben werden mit 0 Punkten bewertet.

Aufgabe 1 - Mockups (25P)

Ziel dieser Aufgabe ist, das Prototyping in Form von Oberflächen-Mockups zu üben. Erstellt aus den Anforderungen zu den zwei Oberflächen SelectHeroScreen und DungeonScreen jeweils ein Mockup. Es darf sich an den Mockups der Vorlesung und Übung orientiert werden.

SelectHeroScreen

Im "SelectHeroScreen" soll der Nutzer einen neuen Helden erstellen oder (später) einen zuvor gespeicherten Held auswählen können.

- Neuen Helden wird über eine Texteingabe ein Name zugewiesen. Über ein Auswahlfeld wird festgelegt, ob sich dieser Held im Normal- oder Hard Mode befindet. Über einen Button wird nach der Eingabe ein neuer Held angelegt und direkt in einen neuen Dungeon geworfen.
- Bereits bestehende Helden werden in einer Liste aufgeführt. Ihre Coins, Stat-Level und Hard bzw. Normal Mode werden mit abgebildet. Wird ein bestehender Held ausgewählt, so wird dieser in einen neuen Dungeon geworfen.

DungeonScreen

Im "DungeonScreen" soll der Nutzer seinen Helden durch einen mit Monster gefüllten Dungeon führen.

- Über einen Angriff- und einen Verteidigungsbutton kann der Spieler die nächste Aktion seines Helden auswählen.
- Das aktuelle Monster sowie dessen Name, aktuelle Kampfhaltung und Lebenspunkte werden angezeigt.
- Der Name des Dungeons sowie dessen Anzahl von Monstern ist ersichtlich. Weiterhin ist zu erkennen, wie viele Monster bereits besiegt worden sind.
- Das Leben und die Coins des Helden werden zusammen mit seinem Namen angezeigt.
- Die jeweiligen Stats des Helden bekommen eine eigene Komponente. Innerhalb dieses Bereiches wird der Name des Stats, dessen aktuelles Level und aktueller Wert angezeigt. Ebenso ist ein Button zum Aufleveln des Stats und die damit verbundenen Kosten sichtbar.

Lege die erstellten Dateien (**.pdf**, **.jpg**, **.png** oder **.svg**) in einem Ordner mit dem Namen "mockups" in deinem Repository ab. Committe und pushe die Änderung abschließend auf den main-Branch.

Bei der Bewertung wird vor allem darauf geachtet, dass auf den Mockups alle Features erkennbar sind. Die Mockups sollten weiterhin einen eindeutigen Prototyp-Charakter besitzen.

Achte darauf, das Repository der aktuellen Hausaufgabe zu verwenden.

Aufgabe 2 - Test-First-Prinzip (75P)

Ziel dieser Aufgabe ist es, das Test-First-Prinzip und Rigorous Testing einmal selbst durchzuführen. Hierzu ist folgende Aufgabenreihenfolge vorgegeben:

- Schritt 1: Tests designen
- Schritt 2: Tests implementieren
- Schritt 3: Methoden implementieren
- Schritt 4: Fehler dokumentieren

Bitte beachte, dass nach jeder Teilaufgabe ein Commit gemacht werden **muss**. Ohne diese können wir die Arbeit nicht nachvollziehen, wodurch diese automatisch mit **0 Punkten** bewertet wird.

1. Neue Methoden

In der ersten Teilaufgabe sollen die Tests und Methoden in den jeweiligen Klassen erstellt, aber noch nicht implementiert, werden. Ziel des ersten Schrittes ist es, die zu testende Struktur aufzubauen. Nach Bearbeitung sollten aus den Tests die zu testenden Methoden ohne Compiler-Anmerkung aufrufbar sein.

Methoden

Erstelle die folgenden Methoden anhand der Methodensignatur.

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

Tests

Erstelle die folgenden Testklassen und Methoden anhand der Methodensignatur.

```
HeroStatUpdateTest::testHeroStatUpdateNormalBehaviour()
HeroStatUpdateTest::testHeroStatUpdateNotEnoughCoins()
HeroStatUpdateTest::testHeroStatUpdateNullStat()
HeroStatUpdateTest::testHeroStatUpdateNullHero()
HeroStatUpdateTest::testHeroStatUpdateNegativePrice()
```

```
HeroEngagesFightTest::testHeroEngagesFightBothDefend()
HeroEngagesFightTest::testHeroEngagesFightBothAttack()
HeroEngagesFightTest::testHeroEngagesFightEnemyDefendHeroAttack()
HeroEngagesFightTest::testHeroEngagesFightEnemyAttackHeroDefend()
HeroEngagesFightTest::testHeroEngagesFightNullHero()
HeroEngagesFightTest::testHeroEngagesFightUnknownStance()
```

```
EvaluateFightTest::evaluateFightEnemyDies()
EvaluateFightTest::evaluateFightEnemySurvives()
EvaluateFightTest::evaluateFightHeroNormalModeHeal()
EvaluateFightTest::evaluateFightHeroHardModeHeal()
EvaluateFightTest::evaluateFightHeroNull()
EvaluateFightTest::evaluateFightEnemyNull()
```

Committe und pushe die Änderung auf den main-Branch, bevor du mit der nächsten Teilaufgabe fortfährst. Benenne möglichst eindeutig, dass die aktuelle Teilaufgabe abgeschlossen ist.

2. Tests implementieren

In der zweiten Teilaufgabe sollen die Tests zu den Methoden aus der vorangegangenen Teilaufgabe implementiert werden. Die Methoden im TemporaryGameController werden hier noch **nicht** implementiert. Folgende Verhaltensweise ist von den Methoden zu erwarten:

heroStatUpdate

> During any time of the game, the player can level up the stats of a hero. Upon clicking the upgrade button, the level rises by one if the hero has enough coins. Like the level, the stat value and cost rise as well and the heros coins decrease.

heroEngagesFight

> While the enemy stands before the hero in an attacking or defending pose, the player can command the hero to attack or defend. Defending reduces the incoming damage of an attack. If both opponents defend, nobody is damaged and if both attack, both take damage. In either case, the LP of the hero or enemy can never be lower than 0.

evaluateFight

> When the epic battle is finished, the enemy may or may not be defeated. If the enemy is still alive, it will change into attacking or defending pose and wait for the heroes next move.
> If the enemy was slain by the hero, some bounty is collected and the next enemy in line is targeted. After defeating an enemy the hero will be fully healed, if his mode was previously set to normal. If is mode was set to hard, the Hero will only be healed if the defeated enemy was the last enemy of the current dungeon.

Folgende Tests sollen auf eine dem auftretenden Fehler angemessene Exception prüfen:

```
HeroStatUpdateTest::testHeroStatUpdateNullStat()
HeroStatUpdateTest::testHeroStatUpdateNullHero()
HeroStatUpdateTest::testHeroStatUpdateNegativePrice()
HeroEngagesFightTest::testHeroEngagesFightNullHero()
HeroEngagesFightTest::testHeroEngagesFightUnknownStance()
EvaluateFightTest::evaluateFightHeroNull()
EvaluateFightTest::evaluateFightEnemyNull()
```

Für das Aufbauen eines Teildatenmodells ist keine eigenständige Methode gefordert. Es wird empfohlen, jede Startsituation eines Testes in dem Rumpf der jeweiligen Testmethode zu implementieren, da sich diese von anderen unterscheiden kann.

Des Weiteren kann es sinnvoll sein, den Ablauf der einzelnen Testmethoden und deren zu testenden Eigenschaften zunächst mittels Kommentaren im Quelltext zu planen (Methodendesign aus Vorlesung 3).

Nach Bearbeitung dieser Teilaufgabe sollten alle Tests bei der Ausführung fehlschlagen, da keine Logik in den Methoden implementiert wurde.

Committe und pushe die Änderung auf den main-Branch, bevor du mit der nächsten Teilaufgabe fortfährst. Benenne möglichst eindeutig, dass die aktuelle Teilaufgabe abgeschlossen ist.

3. Methoden implementieren

In der dritten Teilaufgabe sollen die Methodenrümpfe der Klasse TemporaryGameController implementiert werden. Diese folgen den schriftlichen Beschreibungen aus der vorangegangenen Teilaufgabe.

Nach Bearbeitung dieser Teilaufgabe sollten mehrere oder alle Tests bei der Ausführung durchlaufen. Sollte dies nicht der Fall sein, korrigiert diese Fehler erst in der folgenden Teilaufgabe. Weiterhin sind folgende Hilfestellungen gegeben:

heroStatUpdate - level up

Als Kosten für das Aufleveln eines HeroStats ist das cost-Feld zu wählen. Sind die Kosten zu hoch, wird kein Wert verändert. Nach dem Erhöhen des Levels um 1 sollen die Kosten (cost) und der Wert (value) steigen. Beide Werte können mit einer Konstante (z.B. 1.1) multipliziert werden, um die Nachfolgewerte zu berechnen und zu persistieren. Verursacht gezielte NullPointerException wie in der Vorlesung gezeigt.

heroEngagesFight - fierce battle

Für Angriff und Verteidigung des Enemys sind dessen Attribute zu verwenden. Für den Hero muss das value-Feld aus AttackStat oder DefenseStat ausgelesen werden. Der Schaden eines Angreifers wird dabei um den Verteidigungswert des Verteidigers verringert, sofern dieser auch verteidigt.

Zur Feststellung der nächsten Aktion eines Enemy oder Hero sollen zwei Konstanten (z.B. ”Attacking” und ”Defending”) gewählt werden. Diese könnten dann nach eigenem Ermessen in einer statischen Klasse oder ähnlichem zur Verfügung gestellt werden. Diese werden im Enemy abgelegt oder für den Hero als Argument der Methode übergeben. Verursacht gezielte NullPointerException wie in der Vorlesung gezeigt.

evaluateFight - heal hero, collect bounty or fight on

Nach einem Kampf wird dem Enemy zufällig einer der beiden ”Kampfhaltungen” neu zugewiesen. Sollte der Enemy besiegt worden sein, so wird dessen coins-Feld als Höhe der Belohnung dem Hero übertragen. Die maximale LP eines Heroes ist immer 100. Beim Heilen wird die LP eines Heroes also wieder auf 100 gesetzt. Verursacht gezielte NullPointerException wie in der Vorlesung gezeigt.

Committe und pushe die Änderung auf den main-Branch, bevor du mit der nächsten Teilaufgabe fortfährst. Benenne möglichst eindeutig, dass die aktuelle Teilaufgabe abgeschlossen ist.

4. Fehler dokumentieren

In der vierten Teilaufgabe sollen die konzeptionellen Fehler in Tests oder Logik korrigiert werden. Dabei soll bei jedem gefundenen Fehler ein Kommentar erstellt werden, welcher folgende Punkte beleuchtet:

- Wo trat der Fehler auf?
- Was hat den Fehler verursacht?
- War es ein Konzeptionsfehler im Test oder eine fehlerhafte Implementierung?

Committe und pushe die Änderung abschließend auf den main-Branch.

Bei der Bewertung wird vor allem darauf geachtet, dass die Reihenfolge des Test-First-Prinzips eingehalten wurde.

Achte darauf, das Repository der aktuellen Hausaufgabe zu verwenden.

Anhang

Es folgt eine Auflistung hilfreicher Webseiten und weiteren 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.

IntelliJ

- IntelliJ shortcut cheat sheet:
  `https://resources.jetbrains.com/storage/products/intellij-idea/docs/IntelliJIDEA_ReferenceCard.pdf`