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/ws1920/programming-methodologies/` zu berücksichtigen.

Abgabefrist ist der 14.11.2019 - 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 Korrekteure sichtbar. Nutzt für diese Abgabe bitte ein neues Repository, welches unter folgendem Link anzulegen ist:

`https://classroom.github.com/a/s33wAzZ9`

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 12. Dieses könnt ihr unter folgendem Link herunterladen und installieren:

`https://adoptopenjdk.net/releases.html?variant=openjdk12&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 PMWS1920_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.

```
├── .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.

Aufgabe 1 - Implementierung eines Klassendiagramms (60P)

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

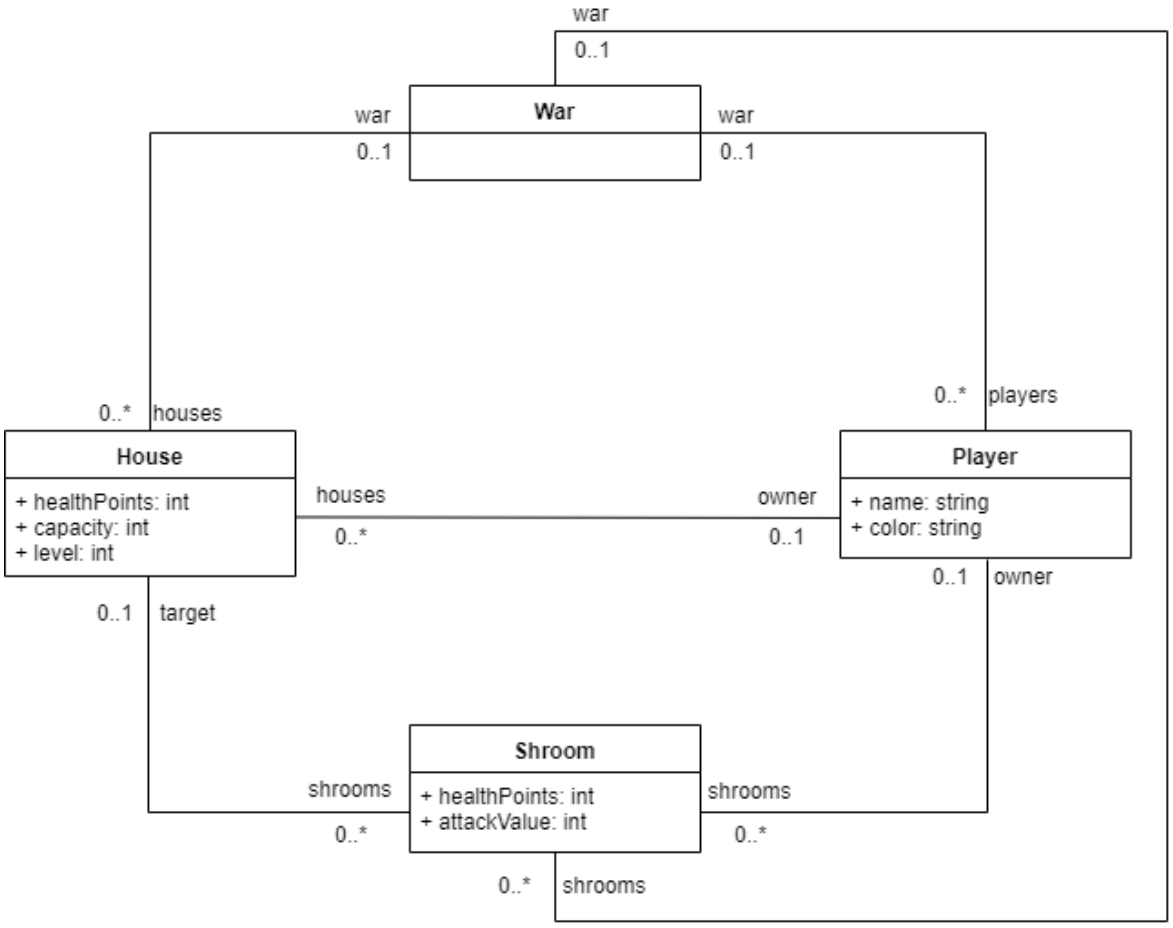

Abbildung 1: Shroom Wars-Klassendiagram (vereinfacht)

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.ws1920.shroomwars.model
2. Füge den Klassen die entsprechenden Attribute hinzu. Achte dabei auf die Zugriffsverkapselung der Attribute durch getter/setter!
3. Es sind keine Konstruktoren notwendig.
4. Implementiere die Assoziationen und stelle die referenzielle Integrität sicher. Dies verlangt folgende Punkte:

- Füge den Klassen für 0...1-Assoziationen die Zugriffsmethoden
  void set<Field> / <Class> get<Field> / <Class> with<Field> hinzu
- Füge den Klassen für 0...*-Assoziationen die Zugriffsmethoden
  void add<Field> / void remove<Field> / <Class> get<Field> / <Class> with<Field> hinzu
- Wähle für die 0...*-Assoziationen eine geeignete Containerklasse (z.B. LinkedHashSet, ArrayList, etc.)
- Sorge dafür, dass bei allen bidirektionalen Assoziationen die Rückrichtung automatisch mitgesetzt wird.
  Beispielsweise, dass beim Setzen eines neuen Hauses für einen Shroom als dessen target, gleichzeitig auch das gewählte Haus den Shroom in seine Collection an Shrooms aufnimmt und (**wichtig!**) der Shroom aus der Collection seines vorherigen Hauses entfernt wird.

Committe und pushe die Änderung an deinem Gradle-Projekt abschließend auf den master-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 Objektdiagramme (Abbildung 3 und 4) ein Test für die Implementierung von Aufgabe 1 entstehen. Hierzu ist folgendes Szenario gegeben:

Title: An army is sent to a Location

Start:	Albert and Martin are playing "Shroom Wars". Martin has 4 shrooms at his-house. Albert has also 4 Shrooms at his house.
Action:	Martin sets Alberts house as his target by clicking 4x on Alberts house.
Result:	All of Martins resting shrooms will now move towards Alberts house by the next tick

Abbildung 2: Szenario von @Gurkensaft

Gradle anpassen

Fügt FulibTools wie in der Vorlesung gezeigt in eure build.gradle-Datei hinzu. FulibTools ist unter folgendem Link zu finden:

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

Nach dem Einfügen der Zeilen sollte der geänderte Teil der Datei wie folgt aussehen.

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

dependencies {
    compile group: 'org.fulib', name: 'fulibTools', version: '1.1.0'
    testCompile group: 'junit', name: 'junit', version: '4.12'

}
```

Im Folgenden kann nun mit der Bearbeitung der Aufgabenstellung begonnen werden.

Für die Implementierung wird folgende Vorgehensweise vorgeschlagen:

1. Erstellt eine Klasse MovementTest.java unter dem Modul src/test/java im zu erstellenden Package de.uniks.ws1920.shroomwars.model.test
2. Erstellt einen Test moveFullArmy(), 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 diagramms/ aus. (Tipp: dieses könnt ihr nun mit Abbildung 3 vergleichen).
5. Implementiert eine neue Methode setTargetFullCharge(), die wie in der Vorlesung beschrieben die Action ausführt, und ruft diese auf 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 diagramms/ aus. (Tipp: dieses könnt ihr nun mit Abbildung 4 vergleichen)

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

Die Diagramme wurden ebenfalls auf dem Blog hochgeladen, schaut sie euch bitte dort an, um alle Einzelheiten in voller Größe anzuschauen.

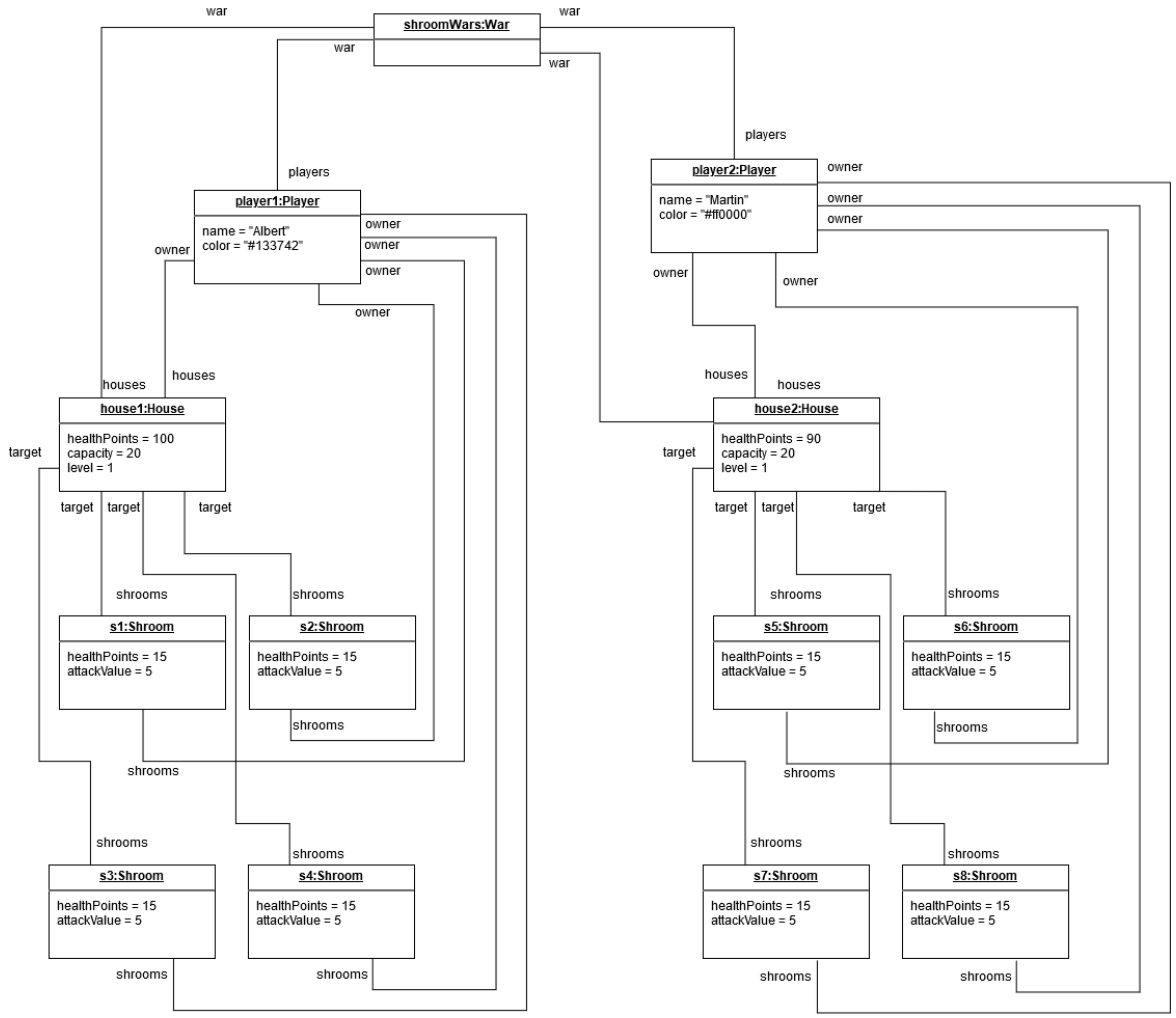

Abbildung 3: Objektdiagramm (Start)

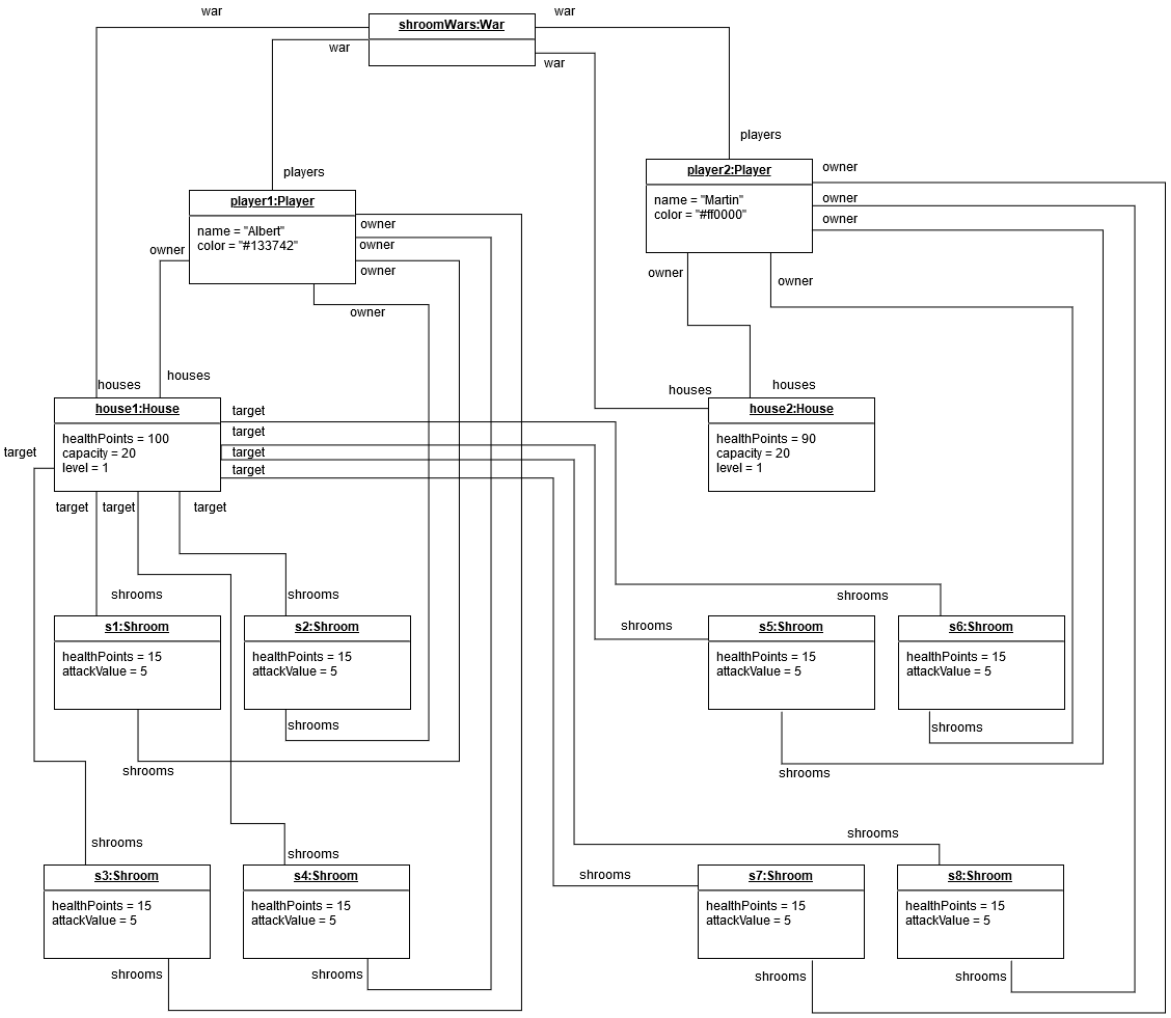

Abbildung 4: Objektdiagramm (Ende)

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.

Sourcetree

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

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`