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 21.11.2019 - 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/X9QUkSjx`

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.

Es ist daher besonders wichtig, die Gitignore (wie in **Übung 3 - Livedemo** gezeigt) vor dem ersten Commit in das Projekt einzufügen.

Aufgabe 1 - Fulib.org (20P)

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 Nutzer-Szenarien darstellen. Die Szenarien der Webseite erfordern keine Unterteilung in Start, Action und End.

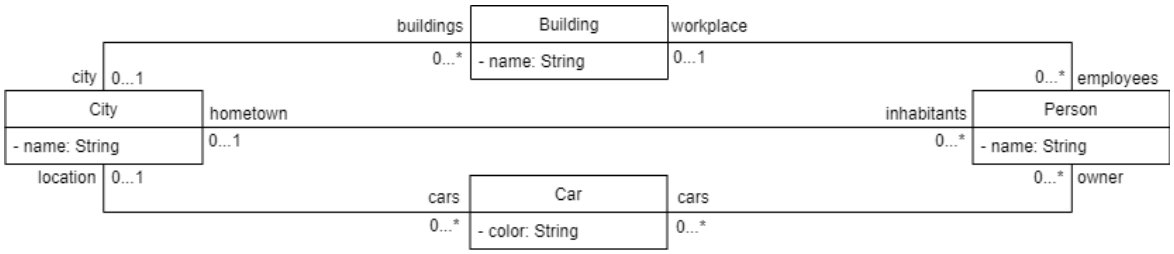

Abbildung 1: Stadt-Klassendiagramm

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

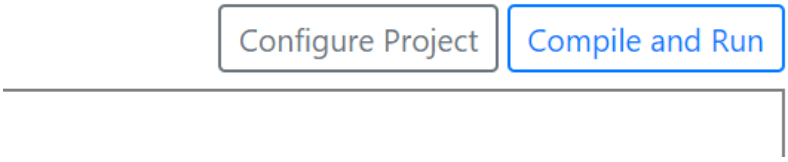

Abbildung 2: Projekt-Button auf Fulib.org

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

Package/Group Name	:	de.uniks.pmws1920
Project Name	:	PMWS1920_ShroomWars_<GitHubName>
Version	:	0.1.0
Scenario File Name	:	Scenario.md

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 master-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 (60P)

In dieser Aufgabe wird das vollständige Datenmodell des Spiels "Shroom Wars" 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.

Füge Fulib wie auf der GitHub-Seite beschrieben zu deinem Gradle-Projekt hinzu.

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

Im Folgenden soll eine Klasse entstehen, die eine Methode besitzt. Diese Methode nutzt die Funktionen von Fulib, um das Datenmodell zu generieren. Hierzu ist das folgende Klassendiagramm gegeben:

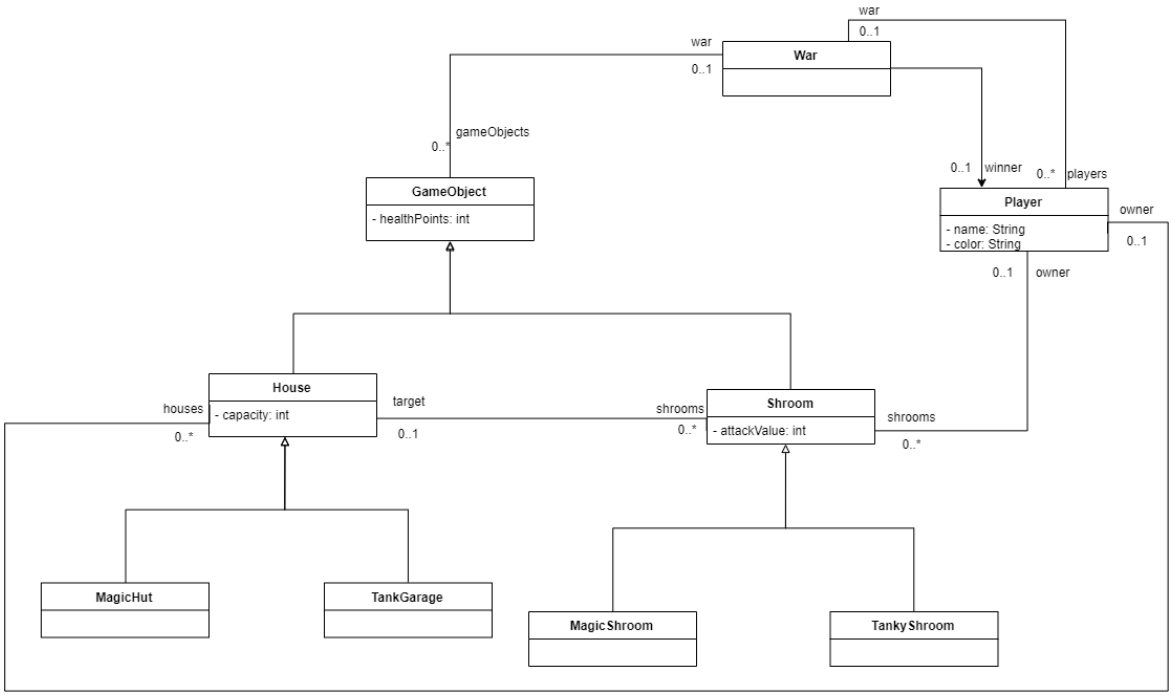

Abbildung 3: „Shroom Wars"-Klassendiagramm

Hinweis:

Die Kanten mit weißen Pfeilköpfen und ohne Bezeichnungen oder Kardinalitäten beschreiben die Vererbung. So erben MagicHut und TankGarage beispielsweise von der Klasse House. Die House-Klasse erbt von GameObject. Für die Shrooms geschieht dies analog.
Die Kante mit ausgefülltem Pfeilkopf vom Game zum Player stellt die einzige unidirektionale Kante im Datenmodell dar.

Folgende Vorgehensweise wird vorgeschlagen. Bitte beachte, dass die Einhaltung der Namenskonventionen hier Pflicht ist:

1. Erstellt die Klasse ModelGen.java unter src/test/java im Package de.uniks.pmws1920.model. Erstelle daraufhin eine Test-Methode mit dem Namen genModel() in der Klasse.
2. Füge die nötigen Zeilen zu dem Test hinzu, die dir erlauben das Package deines generierten Codes festzulegen. Lege dann fest, dass die Dateien in das Package de.uniks.pmws1920.model generiert werden.
3. Füge den Code zur Generierung des gesamten Klassendiagramms zu dem Rumpf der Methode hinzu. (Nutze die Dokumentation und die Beispiele, die auf der GitHub-Seite zur Verfügung gestellt werden, solltest du eine spezifische Methode nicht kennen)
4. Füge eine Zeile hinzu mithilfe der das erstellte Klassendiagram als graphische Datei ausgegeben wird (z.B .svg). Diese Datei soll in einem Ordner im Projekt mit dem Namen diagram abgelegt/generiert werden.
5. Füge die nötigen Zeilen zu dem Test hinzu, die die Generierung der Dateien ermöglichen und generiere dann das Datenmodell.

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 Fulib und die durchgeführte Generierung geachtet.

Achtet darauf, das Repository der aktuellen Hausaufgabe zu verwenden.

Aufgabe 3 - Initialisierung (20P)

In dieser Aufgabe soll das zuvor generierte Datenmodell genutzt werden, um ein initiales Spielgeschehen zu erstellen. Folgende Punkte müssen erfüllt sein:

- Eine Klasse GameController.java unter src/main/java im Package de.uniks.pmws1920.controller muss existieren.
- Diese Klasse muss eine Methode mit dem Namen init() besitzen, die ein War-Objekt zurückgibt. Nur innerhalb dieser Methode dürfen Instanzen des Datenmodells initialisiert werden.
- Das initiale Spielgeschehen muss ein „War"-Spiel mit 2 Spielern enthalten, deren Namen und Farbe (in Worten oder RGB) beliebig gewählt werden dürfen.

  Es müssen 2 Häuser mit einer capacity von 10 und 10 HP erstellt werden, die je einem Spiel zugeordnet sind.

  An jedem dieser Häuser sind 4 (normale) Shrooms zu platzieren, die dem selben Player zugeordnet sind wie das Haus, an dem sie stehen. Jeder Shrooms hat 10 HP und einen attackValue von 2.
- Alle GameObjects müssen mit dem „War"-Spiel verbunden sein.
- Der GameController muss über eine main-Methode ausführbar sein, die die init-Methode aufruft.

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

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

Achtet darauf, das Repository der aktuellen Hausaufgabe zu verwenden.