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 28.11.2019 - 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. Für diese und voraussichtlich alle zukünftigen Abgaben wird kein neues Repository benötigt. Nutzt weiterhin jenes, welches mit folgendem Link angelegt wurde:

`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!

Commit Message Konventionen

In dieser und zukünftigen Hausaufgaben führen wir eine Konvention 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 Abgabe-Commits ein Fehler auffällt, welchen ihr zeitlich gesehen noch korrigieren könnt, so verwendet für den Folge-Commit wieder die Message ”Final HA<Number>”. Der aktuellste Commit mit dieser Nachricht, vor der Deadline, wird als Abgabe gewertet.

Das Ignorieren der Konventionen wird mit 0 Punkten bewertet!

Aufgabe 1 - Test First (50P)

Ziel dieser Aufgabe ist es, das Test-First-Prinzip einmal selbst durchzuführen. Hierzu wird zunächst bereits bestehender Code angepasst und erweitert. Darauf folgt dann die Ausarbeitung der Tests.

1. Umbaumaßnahmen

In diesem Schritt passen wir die bereits bestehende init()-Methode aus Hausaufgabe 4 an. Ändere die Signatur der Methode folgendermaßen:

```
GameController::init(Player... playerList)
```

Die übergebenen Parameter sollen in der Methode anstelle der lokalen Player-Objekte genutzt werden. Wie bereits zuvor, wird das War-Objekt zurückgegeben, sodass in den folgenden Tests damit gearbeitet werden kann.

2. Neue Methoden

Erstelle nun die beiden folgenden Methoden:

```
GameController::upgradeToMagicHut(House house)
GameController::move(House source, House target, int divider)
```

Wie in der Vorlesung gezeigt, kann bereits ein Methodenentwurf durch Kommentare, jedoch **kein Quelltext**, hinzugefügt werden. Die Methoden sollen alle Kombinationsmöglichkeiten der Übergabeparameter sinnvoll (optimalerweise im Sinne der Spielregeln) verarbeiten. Die Mindestanforderung ist dabei, dass die Tests in folgendem Schritt erfolgreich durchlaufen.

3. Test First

In diesem Schritt werden die Tests samt Asserts angelegt und entworfen. Die geplanten Ergebnisse der Tests sind jeweils mit sinnvollen Asserts zu überprüfen.

3.1 TestUpgradeToMagicHut.java

Erstelle die Klasse TestUpgradeToMagicHut.java im Package de.uniks.pmws1920.controller im Modul src/test/java

upgradeToMagicHutNullTest()

Erstelle nun in dieser Klasse die Testmethode:s

`TestUpgradeToMagicHut::upgradeToMagicHutNullTest()`

Rufe darin zunächst die init()-Methode des GameController auf und danach die upgradeToMagicHut()-Methode mit einem null-Wert als Übergabeparameter.

Ergebnis des Tests soll es sein, dass keine Exception ausgelöst wird und sich das Datenmodell nicht verändert.

upgradeToMagicHutScenarioTest()

Erstelle nun in dieser Klasse die Testmethode:

`TestUpgradeToMagicHut::upgradeToMagicHutScenarioTest()`

Rufe darin zunächst die init()-Methode des GameController auf und danach die upgradeToMagicHut()-Methode. Die Übergabeparameter können dem zugehörigen Szenario in Abbildung 1 entnommen werden.

Ergebnis des Tests soll es sein, dass keine Exception ausgelöst wird und sich das Datenmodell wie im Result des Szenarios in Abbildung 1 verändert hat.

Dumpe abschließend ein Objektdiagramm.

3.2 TestMovement.java

Erstelle die Klasse TestMovement.java im Package de.uniks.pmws1920.controller im Modul src/test/java

movementNullTest()

Erstelle nun in dieser Klasse die Testmethode:

`TestMovement::movementNullTest()`

Rufe darin zunächst die init()-Methode des GameController auf und danach die move()-Methode mit null-Werten bzw. -1 als Übergabeparameter.

Ergebnis des Tests soll es sein, dass keine Exception ausgelöst wird und sich das Datenmodell nicht verändert.

movementDivideByZeroTest()

Erstelle nun in dieser Klasse die Testmethode:

`TestMovement::movementDivideByZeroTest()`

Rufe darin zunächst die init()-Methode des GameController auf und danach die move()-Methode mit richtigen Werten für die Häuser, aber 0 als letzten Übergabeparameter.

Ergebnis des Tests soll es sein, dass keine Exception ausgelöst wird und sich das Datenmodell nicht verändert.

movementSzenarioTest()

Erstelle nun in dieser Klasse die Testmethode:

`TestUpgradeToMagicHut::movementSzenarioTest()`

Rufe darin zunächst die init()-Methode des GameController auf und danach die move()-Methode. Die Übergabeparameter können dem zugehörigen Szenario in Abbildung 2 entnommen werden.

Ergebnis des Tests soll es sein, dass keine Exception ausgelöst wird und sich das Datenmodell wie im Result des Szenarios in Abbildung 2 verändert hat.

Dumpe abschließend ein Objektdiagram.

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

Bei der Bewertung wird vor allem darauf geachtet, dass die vollständigen Tests vor der Methoden-Rumpf-Implementation gepusht werden.

Makiert den letzten Commit für diese Teilaufgabe mit "HA5 finished Task1."

Aufgabe 2 - Implementation (50P)

Implementiere nun all die Methoden-Rümpfe der Methoden im GameController.java. Hierbei ist das Ziel, dass nach simultaner Ausführung aller Tests, diese ein positives Ergebnis anzeigen.

Während der Bearbeitung sollen so wenige Änderungen an den Tests wie möglich geschehen. Sollte eine Änderung während der Implementation vorgenommen werden, so muss dies mit einem Kommentar im Quelltext begründet werden.

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

Bei der Bewertung wird vor allem auf die korrekte Umsetzung der Funktionalität und die Ergebnisse der Tests geachtet.

Makiert den letzten Commit für diese Teilaufgabe mit "Final HA5."

Title: Upgrade to Magic Hut

Start:	Alice and Bob are playing “Shroom Wars”. They just initialized a new game. Alice has one house with four shrooms and Bob also has one house with four shrooms.
Action:	Alice upgrades her house to a ”Magic Hut” by sacrificing half of her shrooms at that house.
Result:	Alices has exchanged her house with a new ”Magic Hut” and has 2 shrooms left at the ”Magic Hut”.

Abbildung 1: Upgrade Szenario

Title: Move 50% of the Shrooms

Start:	Alice and Bob are playing “Shroom Wars”. They just initialized a new game. Alice has one house with four shrooms and Bob also has one house with four shrooms.
Action:	Bob moves half of his shrooms from his house to Alice’s house.
Result:	At Bobs house are now 2 shrooms belonging to him. At Alice’s house are 6 shrooms now. 4 of those Shrooms belong to Alice, 2 belong to Bob.

Abbildung 2: Move Szenario

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.

Thema

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