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/ws2223/programming-and-modelling/` zu berücksichtigen.

Abgabefrist ist der 17.11.2022 - 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 benötigst du **ein neues** Repository. Dieses kann über folgenden Link erstellt werden, falls nicht bereits geschehen:

`https://classroom.github.com/a/GCLv72Cn`

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

`https://www.oracle.com/java/technologies/downloads/#java17`

Wählt dort die für euer Betriebssystem passende Datei aus und installiert diese.

Bei der Verwendung einer anderen Java-Version wird diese Hausaufgabe mit 0 Punkten bewertet!

Hinweis

In dieser Hausaufgabe bietet sich die gleichzeitige Bearbeitung beider Aufgaben an. Wie in der Übung gezeigt, werden Test und Implementierung abwechselnd gemäß des Test-First-Prinzip weiterentwickelt. Dieses Vorgehen wird in Zukunft weiter verwendet und kann hier bereits geübt werden.

Bei der Bewertung wird nur der letzte Stand des Repositories vor der Deadline betrachtet.

Aufgabe 1 - Klassendiagramm zu Java (18P)

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

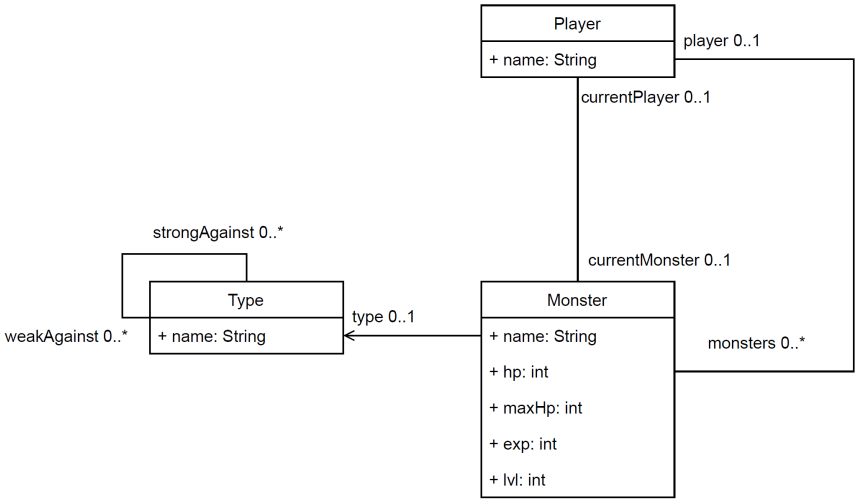

Abbildung 1: PMon-Klassendiagramm (vereinfacht)

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.ws2223.pmon.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>) und <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>() und <Class> set<Field>(<Class> <Variable>) hinzu
   - Füge den Klassen für 0...n-Assoziationen die Zugriffsmethoden List< <Class> > get<Field>() sowie <Class> with<Field>(<Class> <Variable>) und <Class> without<Field>(<Class> <Variable>) hinzu
   - Wähle für die 0...n-Assoziationen eine geeignete Containerklasse (z.B. ArrayList)

Du kannst deine Implementierung mit dem Test aus Aufgabe 2 ausprobieren. Die Datei `.gitkeep` kannst du nach dem Implementieren der Klassen löschen.

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 (10P)

In dieser Aufgabe sollen weitere Tests für Aufgabe 1 implementiert werden. Verwende hierfür die vorgegebene MyModelTest-Klasse in deinem Repository unter src/test/java im Package de.uniks.ws2223.pmon.model. Innerhalb der Klasse müssen folgende Tests implementiert werden:

1. `monsterHp`
2. `monsterCurrentPlayer`
3. `typeWeakAgainst`

Als Orientierung kannst du die bereits vorhandenen Tests in ModelTest verwenden. Beachte, dass die Tests mit sinnvollen Asserts durchgeführt werden müssen.

Aus der `Monster`-Klasse sollen folgende Methoden getestet werden:

1. `setHp`
2. `getHp`
3. `getCurrentPlayer`
4. `setCurrentPlayer`

Beachte, dass `setCurrentPlayer` zusätzlich Effekte auf den Player haben, die anhand von `getCurrentMonster` geprüft werden müssen.

Aus der Klasse `Type` sollen diese Methoden abgedeckt werden:

1. `getWeakAgainst`
2. `withWeakAgainst`
3. `withoutWeakAgainst`

Außerdem sollen Änderungen anhand von `getStrongAgainst` geprüft werden.

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

Bei der Bewertung wird vor allem auf die korrekte Nutzung von JUnit/Assertions geachtet.

Achte darauf, das Repository der aktuellen Hausaufgabe zu verwenden.

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.

IntelliJ IDEA

Bei IntelliJ wird zwischen der kostenlosen „Community“ und der kostenpflichtigen „Ultimate“-Version unterschieden. Es ist für Studierende möglich die „Ultimate“-Version kostenlos zu erhalten, dies ist die „Free Educational License“. Für diese Veranstaltung genügt die kostenlose Version.

- Download: `https://www.jetbrains.com/idea/download/`
- Free Educational Licenses: `https://www.jetbrains.com/community/education/#students`
- Unterschiede der Versionen: `https://www.jetbrains.com/products/compare/?product=idea&product=idea-ce`