Die Hausaufgaben müssen von jedem Studierenden einzeln bearbeitet und abgegeben werden. Die Abgabe muss bis Donnerstag 24.11.2016 um 23:59 Uhr erfolgen. Für die Hausaufgabe sind die aktuellen Informationen vom Blog zu berücksichtigen:
`https://seblog.cs.uni-kassel.de/ws1617/programming-methodologies/`

Alle Abgaben müssen über Gitlab erfolgen:
Link zu unserem GitLab:
`http://avocado.uniks.de:10001`
Zur Erstellung eines Accounts bei unserem GitLab existiert ein Registrierungsservice unter:
`http://avocado.uniks.de:10004`
Als Alternative kann ein Account bei `http://www.gitlab.com` eingerichtet werden. Innerhalb dieses Projektes, müssen anschließend die Betreuer der Übung als Master-Member hinzugefügt werden.

Bei dem Projekt muss die aktuelle SDMLib-pm.jar (Achte auf die Versionsnummern)
`https://repo1.maven.org/maven2/org/sdmlib/SDMLib/`
im Project-Order libs abgelegt und im Build-Path eingetragen werden.

Abgaben per Mail werden nicht mehr akzeptiert.

Diese Hausaufgabe gibt **100** Punkte.

WICHTIG Benennen Sie ihr Projekt für diese und alle zukünftigen Abgaben nach folgendem Schema:

```
PMWS1617_<Matrikelnummer>
```

Der Platzhalter <Matrikelnummer> steht für die Martikelnummer des Studierenden.
Beispiel:

```
PMWS1617_12345678
```

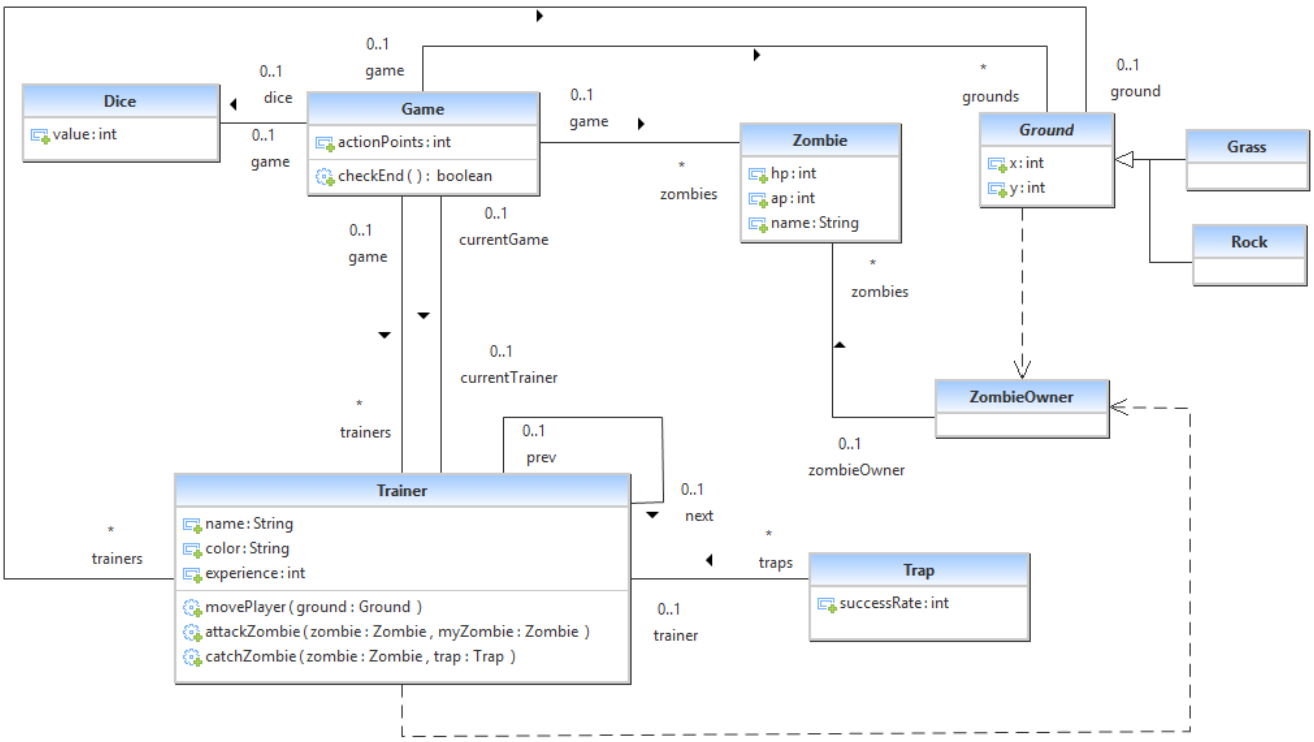

Abbildung 1: ZombieGo Klassendiagram

Aufgabe 1 - Modellierung mit UML Lab (20P)

Gegeben ist das in Abbildung 1 dargestellte Klassendiagramm. Modellieren Sie das Klassendiagramm mit UML Lab. Erzeugen Sie den Quellcode in einem neuen Source-Ordner ”gen”. Dieser sollte nicht im Build-Path eingetragen werden.

Aufgabe 2 - Modellierung mit SDMLib (30P)

Bevor Sie mit der Aufgabe beginnen, entfernen Sie sämtlichen Quellcode aus dem Package ”de.uniks.pm.game.model” im ”src” Ordner. Gegeben ist das in Abbildung 1 dargestellte Klassendiagramm. Modellieren Sie das Klassendiagramm als SDMLib Test. Folgende Vorgehensweise wird allgemein für die Modellierung vorgeschlagen:

1. Erstellen Sie die Testklasse `de.uniks.pm.game.test.GenModelTest.java`
2. Denkt an das hinzufügen der JUnit-Bibliothek
3. Erstellen Sie in der Test-Methode ein ClassModel mit dem Packagenamen ”de.uniks.pm.game.model”.

4. Erstellen Sie für jede Klasse im Diagramm eine Clazz über das ClassModel.
5. Legen Sie über die Methode withAttribute(..) alle Attribute in den Klassen an.
6. Fügen Sie den Klassen über die Methode withMethod(..) die ensprechenden Methoden hinzu.
7. Modellieren Sie die Generalisierungen über die Methode withSuperClass(..).
8. Interfaces können mithilfe der Methode enableInterface() als Interfaces definiert werden.
9. Modellieren Sie die Assoziationen über die Methode withBidirectional(..) und passen Sie die Rollennamen und Kardinalitäten über die zusätzlichen Parameter dem Klassendiagramm entsprechend an.
10. Fügen Sie den Aufruf der Methode generate(”src/main/java”) hinzu, dieser generiert bei Ausführung aus dem Modell Javaquellcode in den jeweiligen Source-Ordner.
11. Fügen Sie den Aufruf der Methode dumpClassDiag(”src”,”classDiag”) hinzu, dieser generiert bei Ausführung ein Klassendiagramm.
12. Starten Sie abschließend die Test-Methode.

Aufgabe 3 - Test (25P)

Nutzen Sie das in Aufgabe 2 erzeugte Modell!

Schreiben Sie einen Test der die in Aufgabe 4 geforderten Bedingungen sinnvoll prüft. Nutzen Sie bitte die Testklasse `de.uniks.pm.game.test.TestModelCreation.java`.

Hinweis: Erstellen Sie die Testklasse im ’src/test/java’ - Ordner.

Aufgabe 4 - Implementierung (25P)

Nutzen Sie das in Aufgabe 2 erzeugte Modell!

Implementieren Sie in der Klasse `de.uniks.pm.game.model.Game` den Rumpf der Methode

```
init(trainers:Trainer...):void
```

- Legen Sie für jeden Trainer jeweils fünf SuperTraps (Trap mit Successrate: 50), eine Mastertrap (Trap mit Successrate: 100), eine Ultratrap (Trap mit Successrate: 125) und zehn Regualartraps (Trap mit Successrate: 10) an.
- Erstellen Sie einen Dice und fügen Sie diesen an das Spielobjekt an.
- Die Koordinaten für Felder werden mit {x,y} definiert. Der Wertebereich der Koordinaten fängt bei 0 an. Siehe Definition in dem Listing 1.
  - Die Zahlen 24,25,26,47,48,49,70,71,72 sind als Grassfelder zu modellieren.
  - Die Rockfelder werden für die Zahlen 74 erstellt.
  - Die Felder mit dem Index 0 sind leere Felder
- Setzen Sie die `nextTrainer` und `previousTrainer` Kanten. (Letzter <-> Erster nicht vergessen)
- Der erste Trainer soll auf das Grassfeld mit den Koordinaten (0,0) im Spiel gesetzt werden.
- Der zweite Trainer soll auf das Grassfeld mit den Koordinaten (4,5) im Spiel gesetzt werden.
- Der erste Trainer soll am Zug sein.
- Beim Game sollen drei Action Points zur Verfügung stehen.

Wenn Sie fertig sind sollte der Test aus Aufgabe 3 ohne Fehler laufen.

```
<map version="1.0" orientation="orthogonal" renderorder="right-down"
  width="5" height="6" tilewidth="32" tileheight="32">
  <layer name="Background" width="10" height="10">
  <data encoding="csv">
    24,25,25,25,26,
    47, 0,48, 0,49,
    47,48,48,48,49,
    47,48, 0,74,49,
    47,48,48,48,49,
    70,71,71,71,72
  </data>
 </layer>
</map>
```

Listing 1: TMXDatei