Die Hausaufgaben müssen von jedem Studierenden einzeln bearbeitet und abgegeben werden. Die Abgabe muss bis Freitag 23.11.2016 um 13:00 Uhr erfolgen. Für die Hausaufgabe sind die aktuellen Informationen vom Blog zu berücksichtigen `https://seblog.cs.uni-kassel.de/ws1819/programming-methodologies/` und aus den Übungen zu berücksichtigen.

Abgaben per Mail werden nicht akzeptiert. Alle Abgaben müssen über GitHub erfolgen:
Link zu unserem GitHub:
`https://classroom.github.com/a/NtisvTmL`
Diese Hausaufgabe gibt **100** Punkte.

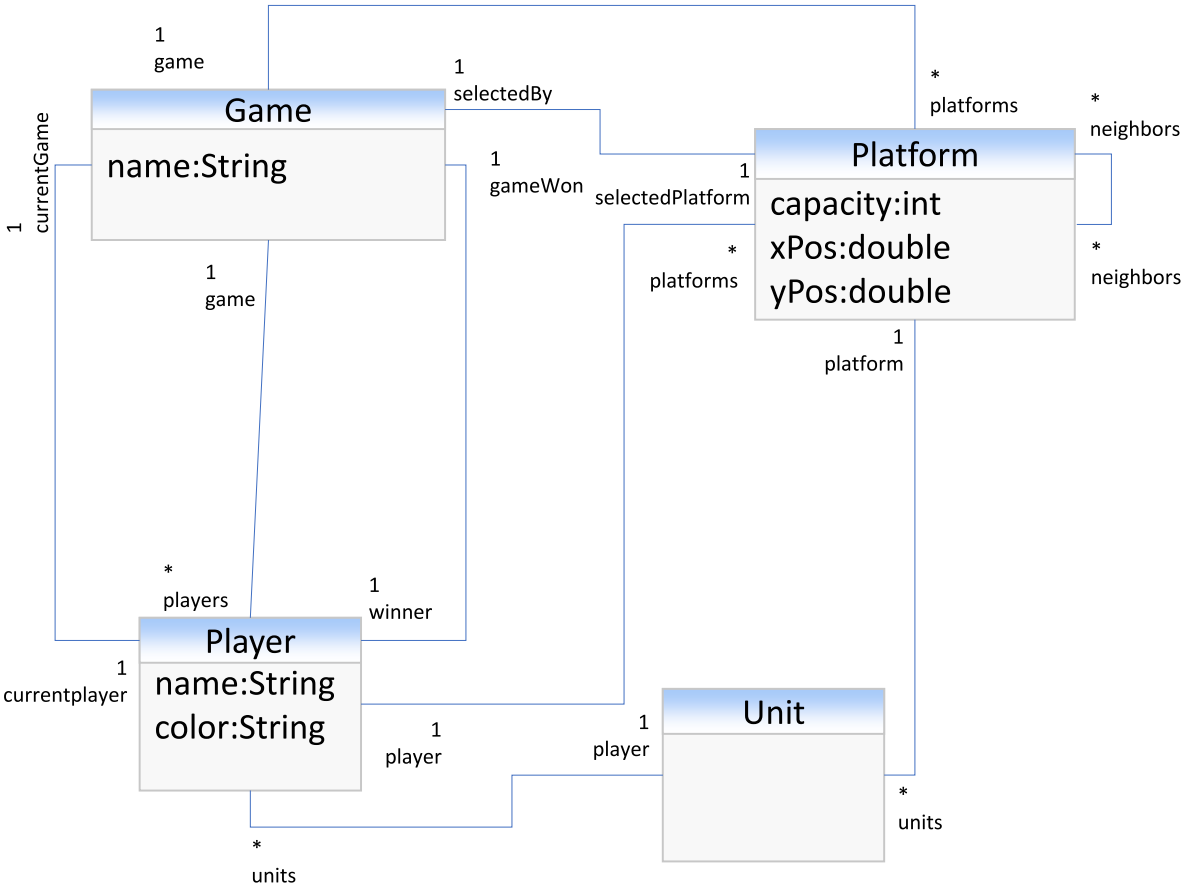

Abbildung 1: LiveRisk Klassendiagramm

Aufgabe 0 - Vorbereitung (1P)

Vorbereitung für die Hausaufgabe.

1. Erstelle ein Gradle-Project
2. Verknüpfe das Projekt mit deiner Assignment 4 auf GitHub
3. Füge JUnit-Bibliothek zu Gradle hinzu. (Falls noch nicht geschehen)

Aufgabe 1 - Modellierung mit NetworkParser (25P)

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

```
repositories {
  maven {url 'https://oss.sonatype.org/content/repositories/snapshots'}
}
dependencies {
  compile group:"de.uniks",name: "NetworkParser",
    version: "latest.integration",classifier:"sources18",changing: true
}
```

Listing 1: Gradle

1. Füge NetworkParser zu Gradle hinzu.
2. Erstellen Sie die Testklasse `de.uniks.liverisk.test.GenModelNetworkParserTest.java` im ”src/test/java”
3. Erstellen Sie in der Test-Methode ein ClassModel mit dem Packagenamen ”de.uniks.liverisk.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. Modellieren Sie die Assoziationen über die Methode withBidirectional(..) und passen Sie die Rollennamen und Kardinalitäten über die zusätzlichen Parameter dem Klassendiagramm entsprechend an.
7. Fügen Sie den Aufruf der Methode generate(”gen-np”) hinzu, dieser generiert bei Ausführung aus dem Modell Javaquellcode in den jeweiligen Source-Ordner.
8. Fügen Sie den Aufruf der Methode dumpHTML("gen-np/classDiagtest.html", true) hinzu, dieser generiert bei Ausführung ein Klassendiagramm.
9. Starten Sie abschließend die Test-Methode.

Aufgabe 2 - Modellierung mit fulib (25P)

Gegeben ist das in Abbildung 1 dargestellte Klassendiagramm. Modellieren Sie das Klassendiagramm als fulib Test. Folgende Vorgehensweise wird allgemein für die Modellierung vorgeschlagen:

1. Füge fulib zu Gradle hinzu.
2. Erstellen Sie die Testklasse `de.uniks.liverisk.test.GenModelTest.java` im "src/test/java"
3. Erstellen Sie in der Test-Methode ein ClassModelBuilder mit dem Packagenamen "de.uniks.liverisk.model" und SourceFolder "src/main/java".
4. Erstellen Sie für jede Klasse im Diagramm eine ClassBuilder über das ClassBuilder.
5. Legen Sie über die Methode buildAttribute(..) alle Attribute in den Klassen an.
6. Modellieren Sie die Assoziationen über die Methode buildAssociation(..) und passen Sie die Rollennamen und Kardinalitäten über die zusätzlichen Parameter dem Klassendiagramm entsprechend an.
7. Fügen Sie den Aufruf der Methode getClassModel hinzu.
8. Fügen Sie den Aufruf der Methode Fulib.generator().generate hinzu, dieser generiert bei Ausführung aus dem Modell Javaquellcode in den jeweiligen Source-Ordner.
9. Starten Sie abschließend die Test-Methode.

```
1 repositories {
2   maven {url 'https://oss.sonatype.org/content/repositories/snapshots'}
3 }
4 dependencies {
5    testCompile 'org.fulib:fulib:1.0.+'
6 }
```

Listing 2: Gradle

Aufgabe 3 - Test (28P)

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.liverisk.test.TestModelCreation.java`.

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

Aufgabe 4 - Implementierung (20P)

Nutzen Sie das in Aufgabe 2 erzeugte Modell!

Implementieren Sie in der Klasse `de.uniks.liverisk.controller.GameController` den Rumpf der Methode

```
init(player:Player ...) : Game
```

- Erstellen Sie eine Game Instanz
- Fügen Sie alle Player zum Game hinzu
- Erstellen sie für jeden Spieler zusätzlich 5 Units
- Legen sie mindestens 4 Plattformen an
- Die Koordinaten für Plattformen werden mit {x,y} definiert.
  Die Koordinaten der Plattformen sind [{50, 200}, {650, 200}, {300, 100}, {400,300}]
  Die Capacity der Plattformen wird durch das Diagramm 2 definiert.
- Verknüpfen Sie alle Plattformen mit der Game Instanz
- Setzen sie die Kanten zwischen den Plattformen wie in dem Diagramm 2 dargestellt
- Setzen sie die Kanten zwischen dem Spieler und seinen Plattformen
- Setzen sie die jeweils erste Einheit des Spielers auf seine Startplattform
- Der erste Spieler soll am Zug sein

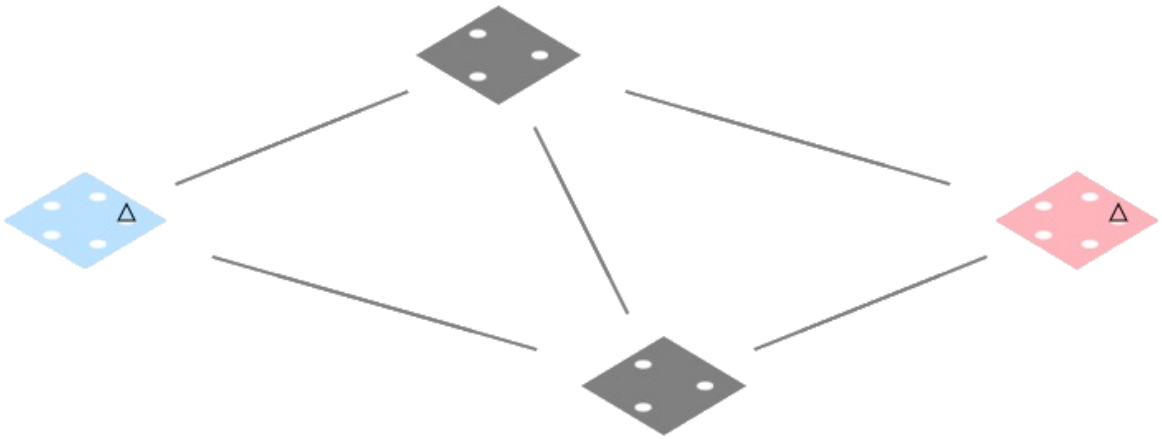

Abbildung 2: Anfangszustand LiveRisk

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

Aufgabe 4 - Evaluation (1P)

Bitte füllen Sie den Fragebogen unter
`https://goo.gl/forms/6IDiKD8oyCW9OpAA2` vollständig aus.