Diese Hausaufgabe stellt eine Bonusleistung zu den Hausaufgaben dar. Sie ist **keine** Pflichtabgabe. Bei einer erfolgreichen Bearbeitung kann eine nicht bestandene Hausaufgabe (weniger als 50% der Punkte) ausgeglichen werden. Die Hausaufgabe muss alleine bearbeitet und abgegeben werden. Für die Hausaufgabe sind die aktuellen Informationen vom Blog `https://seblog.cs.uni-kassel.de/ws2021/programming-and-modelling/` zu berücksichtigen.

Abgabefrist ist der 04.03.2021 - 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 benötigt ihr ein neues Repository, welches über folgenden Link angelegt werden kann:

`https://classroom.github.com/a/5tB3K7_e`

Nicht oder zu spät gepushte (Teil-)Abgaben werden nicht 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.
Zur Abgabe der Hausaufgaben darf ein beliebiges Git-Tool genutzt werden

Gitignore

Füge eine Gitignore zu deinem Projekt hinzu. Diese muss im Root-Verzeichnis des Projektes liegen (also parallel zu dem .git-Verzeichnis) und vor dem ersten Commit eingebunden sein. **Das Nicht-Einbinden der Gitignore führt zu Punktabzug.**

Projekt initialisieren

Es ist ratsam das geforderte Gradle Projekt vor der ersten Aufgabe zu initialisieren, um die zu erstellenden Dateien wie vorgegeben ablegen zu können.

Das Gradle Projekt **muss** den Namen PMWS2021_Bonus_<Githubname> tragen.

Lest vor dem Beginn der Bearbeitung alle Aufgaben sorgfältig durch!

Aufgabe 1 - Konzipierung

In dieser Aufgabe wird die grundlegende Konzipierung eurer Anwendung durchgeführt. Es soll das Spiel Schere-Stein-Papier und dessen erweiterte Form Schere-Stein-Papier-Echse-Spock programmiert werden.

Im folgenden Diagramm können die Spielregeln der erweiterten als auch klassischen Variante abgelesen werden:

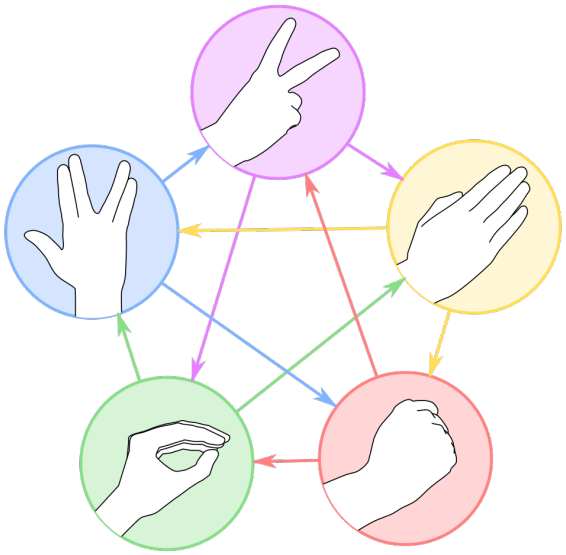

Abbildung 1: Regeln von Schere-Stein-Papier-Echse-Spock

Falls Ihr zu den Spielen weitere Informationen braucht, könnt ihr diese unter folgendem Link nachlesen: `https://de.wikipedia.org/wiki/Schere,_Stein,_Papier`.

Hinweise zur Abgabe:
Folgende Dateitypen sind für die Abgabe von Aufgabe 1 erlaubt: .txt, .png, .jpg, .svg, .pdf.

Handschriftliche Abgaben werden nicht gewertet.

1.1 User-Szenarien

Für die erste Aufgabe sollen zwei textuelle Szenarien zu konkreten Spielsituationen geschrieben werden. Die Szenarien sind, wie aus Hausaufgabe 1 bekannt, auf Englisch zu verfassen und in vier Abschnitte zu unterteilen: Title, Start, Action, End.

Das Szenario aus Aufgabe 1.2 darf für diese Aufgabe nicht verwendet werden. Es müssen zwei von euch neu geschriebene Szenarien entstehen.

Lege die zwei erstellten Dateien in einem Ordner mit dem Namen task1.1 im Root-Verzeichnis deines Projektes ab.

1.2 Objektdiagramme

Leitet für das folgende User-Szenario und für die **beiden** User-Szenarien aus der vorangegangenen Aufgabe Objektdiagramme ab. Erstellt dazu jeweils ein Objektdiagramm zur Start- und Endsituation. Für jedes der drei Szenarien müssen somit zwei Diagramme entstehen. Benennt die Dateien eindeutig (beispielsweise <Szenariotitel><Start | End>).

Title: Lose a match

Start: Alice and Bob are playing RPS. They are playing with the extension Lizard and Spock. The rules are set to best of three. Alice and Bob have both won one round each. It's the third round.

Action: Alice is choosing Spock.

End: Bob chose Lizard. Lizard beats Spock. Alice loses the third round. Bob has won 2 rounds in total. Bob wins this match (best of three).

Legt die **sechs** erstellten Dateien in einem Ordner mit dem Namen task1.2 im Root-Verzeichnis deines Projektes ab.

1.3 Klassendiagramm

Erstellt ein Klassendiagramm, welches sämtliche Objektdiagramme aus Aufgabe 1.2 vereinigt.

Legt die erstellte Datei in einem Ordner mit dem Namen task1.3 im Root-Verzeichnis deines Projektes ab.

Aufgabe 2 - Umsetzung

Ziel dieser Aufgabe ist es, eine voll funktionsfähige Anwendung zu erstellen.

Aufgabe 2.1 Projektvorgaben

Es folgen Vorgaben zu den zu verwendenden Technologien, sowie der Projektstruktur:

- Für die Oberflächen muss JavaFx verwendet werden.
- Die Anwendung muss mittels MVC Pattern implementiert werden.
  - Das Modell wird in das Package de.uniks.pmws2021.rps.model generiert.
  - FXML-Dateien werden in src/main/resources im Package de.uniks.pmws2021.rps.view abgelegt.
  - Controller werden im Package de.uniks.pmws2021.rps.controller abgelegt.
- Die Modellgenerierung muss mit Fulib oder FulibScenarios durchgeführt werden.
- Analog zu HA07 müssen alle schreibenden Zugriffe auf das Datenmodell über einen Editor geschehen. Dieser heißt nun RPSEditor und ist im Package de.uniks.pmws2021.rps abzulegen.
- Analog zu HA06 müssen alle Szenenwechsel über einen StageManager geschehen. Dieser ist im Package de.uniks.pmws2021.rps abzulegen.

Aufgabe 2.2 Oberfläche und Funktionalität

Anhand der folgenden Mockups wird die geforderte Funktionalität eurer Anwendung verdeutlicht. Die Mockups sind erneut nur als Vorlagen gedacht, ihr könnt das Layout eurer Anwendung beliebig ändern. Eure Anwendung muss drei separate Screens beinhalten.

Wichtig zu beachten:

Wenn beide Spieler das gleiche Symbol auswählen, wird die Runde wiederholt.

Aufgabe 2.2.1 StartScreen

Im StartScreen, dem Startpunkt eurer Anwendung, wird neben dem Spielmodus auch entschieden, wie viele Runden gespielt werden sollen.

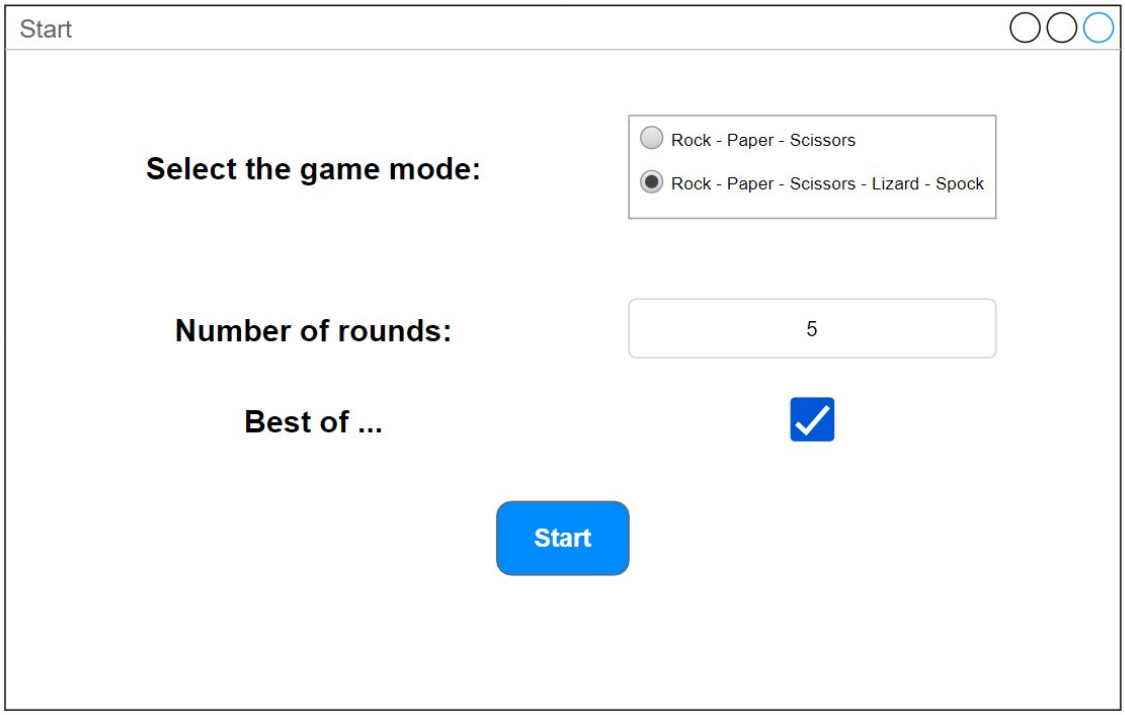

Abbildung 2: StartScreen

Funktionalität:

- Es muss eine Auswahlmöglichkeit für den Spielmodus geben
- Es muss mittels Eingabefeld möglich sein, die Anzahl der Runden festzulegen
- Neben der Rundenanzahl muss es möglich sein, die Option Best of... zu aktivieren. Erklärungen zum Best of...-Modus:
  - Rundenanzahl ist immer ungerade
  - Ein Match kann nur gewonnen oder verloren werden
  - Nachdem ein Spieler z.B. 2/3 Runden gewonnen hat, endet das Spiel
- Ein Start Button initialisiert das Spiel mit den gesetzten Optionen und wechselt zum InGameScreen

Aufgabe 2.2.2 IngameScreen

Im IngameScreen werden anhand der im StartScreen getätigten Optionen ein Spiel aufgebaut.

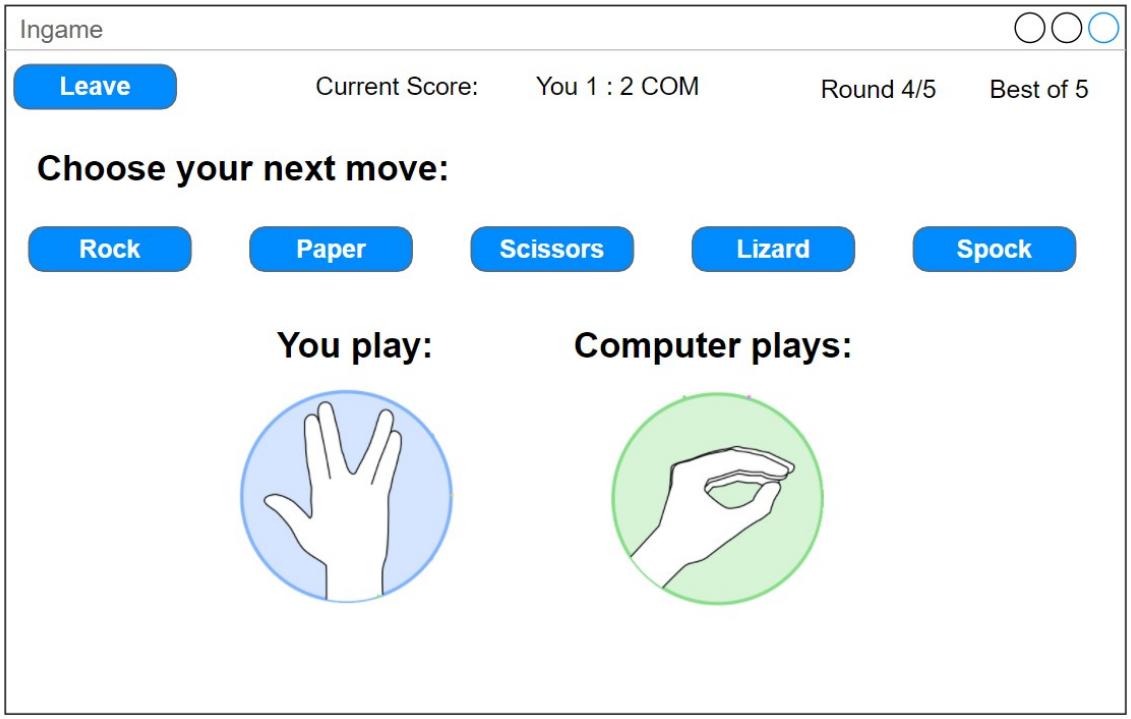

Abbildung 3: IngameScreen

Funktionalität:

- Der Leave-Button bricht das Spiel ab und wechselt zum StartScreen
- Es wird gezählt, wie viele Runden jeder Spieler gewonnen hat
- Dem Spieler wird neben der aktuell gespielten Runde ebenfalls die maximale Anzahl der Runden angezeigt
- Entsprechend des Spielmodus muss es dem Spieler möglich sein, ein Symbol auszuwählen
- In jeder Runde werden die ausgewählten Symbole von Spieler und Computer angezeigt. Diese müssen als Bild angezeigt werden
- Es wird immer gegen einen Computer gespielt. Also einen Bot, welcher zufällig ein Symbol auswählt

Aufgabe 2.2.3 ResultScreen

Nachdem ein Spiel beendet wurde und ein finales Ergebnis vorliegt, wird der ResultScreen geöffnet.

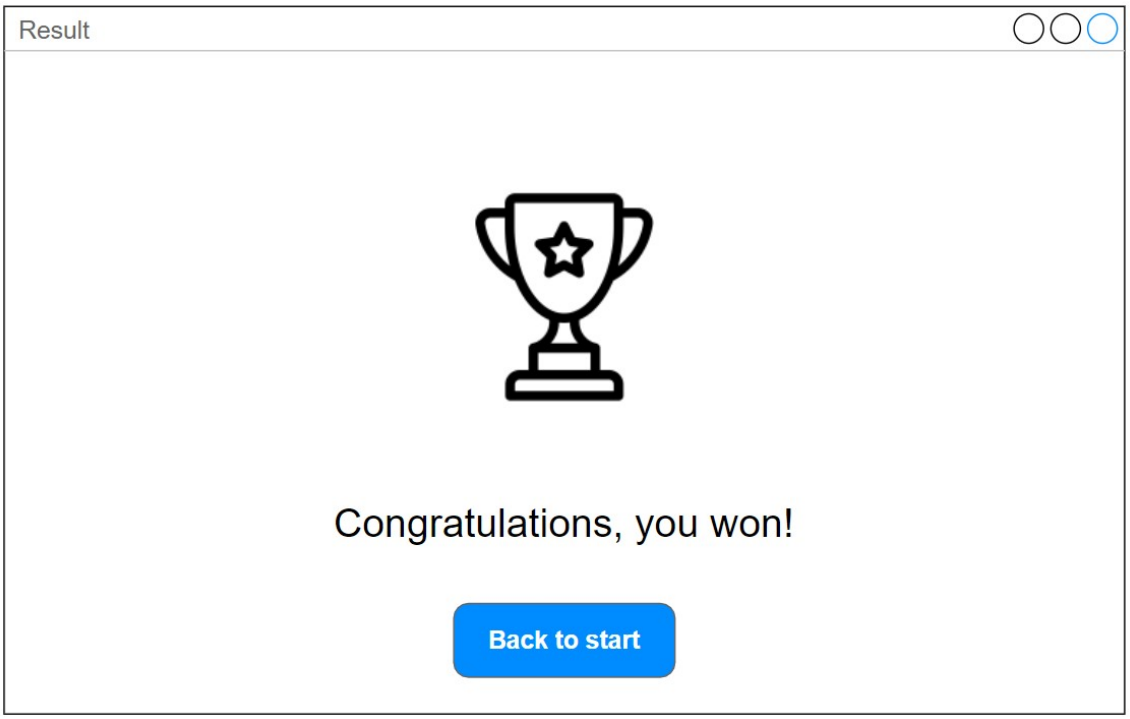

Abbildung 4: ResultScreen

Funktionalität:

- Dem Spieler wird angezeigt, wie das Spiel geendet hat. Dies muss mindestens durch ein Bild und einen Text geschehen
- Back to Start-Button wechselt zum StartScreen

Aufgabe 3 - Tests

In dieser Aufgabe soll die zuvor entstandene Anwendung mit JUnit getestet werden. Hierfür müssen folgende Tests erstellt werden.

Aufgabe 3.1 TestGameModeSelection.java

In dieser Klasse sollen die optischen Veränderungen des IngameScreens getestet werden, welche auf der Wahl des Spielmodus beruhen.

Aufgabe 3.1.1 testClassicModeGameStart()

Dieser Test soll ein Match im klassischen Schere-Stein-Papier Modus starten. Es muss überprüft werden, ob nach dem Wechsel in den IngameScreen alle zu erwartenden Elemente angezeigt werden.

Aufgabe 3.1.2 testExtendedModeGameStart()

Dieser Test soll ein Match im erweiterten Schere-Stein-Papier-Echse-Spock Modus starten. Es muss überprüft werden, ob nach dem Wechsel in den IngameScreen alle zu erwartenden Elemente angezeigt werden.

Aufgabe 3.2 TestPlayModeSelection.java

In dieser Klasse sollen die optischen Veränderungen des IngameScreens getestet werden, welche auf der Wahl der Gewinnbedinung beruhen.

Aufgabe 3.2.1 testThreeRoundNormalConditionGameStart()

Dieser Test soll ein Match im klassischen Schere-Stein-Papier Modus mit 3 Runden starten. Es muss überprüft werden, ob nach dem Wechsel in den IngameScreen alle zu erwartenden Elemente angezeigt werden.

Aufgabe 3.2.2 testThreeRoundBestOfConditionGameStart()

Dieser Test soll ein Match im klassischen Schere-Stein-Papier Modus in Best of 3 starten. Es muss überprüft werden, ob nach dem Wechsel in den IngameScreen alle zu erwartenden Elemente angezeigt werden.

Aufgabe 3.3 TestFullMatch.java

In dieser Klasse sollen vollständige Spieldurchläufe simuliert werden. Die Testmethoden sollen dabei Veränderungen an der Oberfläche, das Einhalten der Gewinnbedingung, sowie das Einhalten der Spielregeln prüfen.

Aufgabe 3.3.1 testWinBestOfThreeFullGame()

Dieser Test soll ein Match im klassischen Schere-Stein-Papier Modus in Best of 3 starten. Das begonnene Spiel muss folgende Spielzüge umfassen:

Player		**Computer**
Schere	-	Stein
Papier	-	Stein
Stein	-	Schere

Es muss geprüft werden, ob während dem Verlauf des Matches die Änderungen an Oberfläche und Datenmodell korrekt sind. Abschließend muss geprüft werden, ob der ResultScreen das zu erwartende Ergebnis des Spiels anzeigt.

Aufgabe 3.3.2 testLoseFiveRoundFullGame()

Dieser Test soll ein Match im erweiterten Schere-Stein-Papier-Echse-Spock Modus mit 5 Runden starten. Das begonnene Spiel muss folgende Spielzüge umfassen:

Player		**Computer**
Spock	-	Echse
Echse	-	Schere
Papier	-	Echse
Spock	-	Papier
Schere	-	Stein

Es muss geprüft werden, ob während dem Verlauf des Matches die Änderungen an Oberfläche und Datenmodell korrekt sind. Abschließend muss geprüft werden, ob der ResultScreen das zu erwartende Ergebnis des Spiels anzeigt.

Aufgabe 3.3.3 testDrawWithDrawRound()

Dieser Test soll ein Match im klassischen Schere-Stein-Papier Modus mit 2 Runden starten. Das begonnene Spiel muss folgende Spielzüge umfassen:

Player		**Computer**
Stein	-	Stein
Papier	-	Schere
Stein	-	Schere

Es muss geprüft werden, ob während dem Verlauf des Matches die Änderungen an Oberfläche und Datenmodell korrekt sind. Abschließend muss geprüft werden, ob der ResultScreen das zu erwartende Ergebnis des Spiels anzeigt.