Das Projekt muss von jedem Studierenden einzeln bearbeitet und abgegeben werden. Für das Projekt sind die aktuellen Informationen vom Blog https://seblog.cs.uni-kassel.de/ws2324/programming-and-modelling/ zu berücksichtigen.

Abgabefrist ist der 25.02.2024 – 23:59 Uhr

Abgabe

Für das Projekt benötigst du ein **neues** Repository. Dieses kann über folgenden Link eingesehen oder auch erstellt werden:

https://classroom.github.com/a/SrLm5Pom

Selbstständigkeitserklärung

In dem Repository gibt es eine Selbstständigkeitserklärung. Diese **muss** zur Deadline des Projektes ausgefüllt und **unterschrieben** im Repository hinterlegt sein.

Vorbereitung

Mach dich mit dem Spiel „Mensch ärgere dich nicht" vertraut. Die Anleitung findest du hier:

https://de.wikipedia.org/wiki/Mensch_%C3%A4rgere_Dich_nicht#Anleitung

Die dort aufgeführten optionalen Regeln können ignoriert werden.

Als Spielbrett soll das klassische Spielbrett für 4 Spieler umgesetzt werden. Entsprechend soll es zwei bis vier Personen ermöglicht werden, gemeinsam vor dem Computer mit dem zu entwickelnden Programm „Mensch ärgere dich nicht" zu spielen.

Auf den folgenden Seiten werden die zu bearbeitenden Aufgaben beschrieben. Das Programm darf in soweit frei gestaltet, mit zusätzlicher Funktionalität versehen und zusätzlich getestet werden, sodass die dargelegten Anforderungen erfüllt sind. Durch zusätzliche Funktionalität können allerdings keine Bonuspunkte erzielt werden.

Wir empfehlen, regelmäßig zu committen und zu pushen. Es gibt keine Vorgaben für Commit-Messages und zur Bearbeitungsreihenfolge. Dein Projekt muss vor der Deadline auf den main-Branch gepusht worden sein.

Aufgabe 1 – Modellierung

Im Laufe des Projekts soll das Spiel „Mensch ärgere dich nicht" modelliert und programmiert werden, sodass als Endprodukt ein Programm ensteht, das es zwei bis vier Spielern ermöglicht, gegeneinander „Mensch ärgere dich nicht" über eine grafische Oberfläche zu spielen.

Hierzu sollen zunächst die kennengelernten Methoden zur Modellierung angewendet werden.

Achte auf die vorgegebenen Dateiformate. Von Hand gezeichnete Diagramme oder Wireframes werden nicht akzeptiert. Scenebuilder-Screenshots werden als Wireframes nicht akzeptiert.

Hinweis: Das Spielfeld darf **nicht** als multidimensionales Array implementiert werden, sondern muss aus Objekten bestehen, die die benötigten Beziehungen zueinander haben. Es ist ebenfalls **nicht** gestattet, Indizes als Nummer- oder String-Attribut zu kodieren.

Aufgabe 1.1 – Szenarien

Schreibe drei Szenarien zu „Mensch ärgere dich nicht". Die Szenarien müssen in Englisch verfasst sein.

Die Szenarien sollen in deinem Repository im Ordner `modelling` in der Datei `scenarios.md` zu finden sein.

Aufgabe 1.2 – Objektdiagramme ableiten

Leite aus deinen Szenarien Objektdiagramme für Start- und Endsituation ab. Lege die sechs erstellten pdf-Dateien wie folgt in deinem Repository ab:

```
modelling/diagrams/<Szenario-Titel>_<Start bzw. Result>.pdf
```

Aufgabe 1.3 – Klassendiagramm ableiten

Leite aus deinen Objektdiagrammen ein Klassendiagramm ab. Lege die erstellte pdf-Datei wie folgt in deinem Repository ab:

```
modelling/diagrams/classdiagram.pdf
```

Aufgabe 1.4 – Wireframes

Erstelle Wireframes für einen IngameScreen, einen SetupScreen und einen GameOverScreen deines Spiels. Der IngameScreen enthält das Spielfeld, der SetupScreen soll es ermöglichen, ein neues Spiel mit den eingegebenen Spielernamen zu starten, und der GameOverScreen zeigt am Ende eines Spiels an, wer gewonnen und verloren hat.

Lege die pdf-Dateien wie folgt in deinem Repository ab:

```
modelling/wireframes/Ingame.pdf
```

```
modelling/wireframes/Setup.pdf
```

```
modelling/wireframes/GameOver.pdf
```

Aufgabe 2 – Programmierung

In dieser Aufgabe wird das „Mensch ärgere dich nicht"-Spiel vervollständigt. Dazu wird die Oberfläche erstellt und die Spiellogik implementiert. Dem MVC-Pattern entsprechend sollen Oberfläche und Modell(-Logik) durch Controller miteinander verknüpft werden. Außerdem soll durch Tests sichergestellt werden, dass dein Programm funktioniert.

Die Anwendung muss unter Verwendung von JavaFX umgesetzt werden.

Aufgabe 2.1 – Datenmodell generieren

Verwende die bereits in deinem Projekt existierende Klasse GenModel, um dein Klassendiagramm aus Aufgabe 1.3 zu definieren sowie zu generieren.

Hinweis: Wenn du während der Programmierung merkst, dass sich dein Modell ändern muss, kannst du diese Änderungen vornehmen, ohne Aufgabe 1 anpassen zu müssen.

Aufgabe 2.2 – FXML-Dateien

Erstelle anhand der in Aufgabe 1 erstellten Wireframes die entsprechenden FXML-Dateien mit dem Scenebuilder.

Lege diese unter src/main/resources im Ordner de/uniks/pmws2324/ludo/view ab.

Aufgabe 2.3 – MVC

Implementiere folgende Klassen:

- `de.uniks.pmws2324.ludo.Main` und `de.uniks.pmws2324.ludo.App`
  zum Starten der Anwendung und Verwalten von Controllern
- `de.uniks.pmws2324.ludo.service.GameService`
  enthält benötigte Logik-Methoden für das Spiel
- `de.uniks.pmws2324.ludo.Constants`
  als zentraler Ort für Konstanten
- `de.uniks.pmws2324.ludo.controller.SetupController`
  als Controller für den Setup-Screen
- `de.uniks.pmws2324.ludo.controller.IngameController`
  als Controller für den Ingame-Screen
- `de.uniks.pmws2324.ludo.controller.GameOverController`
  als Controller für den GameOver-Screen
- `de.uniks.pmws2324.ludo.controller.<THING>SubController`
  als benötigte(r) Sub-Controller

Danach sollte deine Anwendung **vollständig** ausführbar und spielbar sein. Starte sie mit der Klasse Main, um das Spiel auszuprobieren und zu spielen, bis ein Sieger feststeht.

Aufgabe 2.4 – Tests

Lege für drei selbstgewählte Logik-Methoden[1] des `GameServices` je eine eigene Test-Klasse an. Erstelle in jeder Test-Klasse je drei Test-Methoden, die die Funktionalität der Logik-Methode durch sinnvolles Testen prüfen. Erkläre in jeder Methode in Form von Kommentaren, welche Anforderung an das Verhalten der Logik-Methode geprüft wird. Lege die Test-Klassen im Modul `src/test/java` im package `de.uniks.pmws2324.ludo.service` ab.

Implementiere außerdem einen GUI-Test. Verwende hierfür die Klasse `FullGameTest` im Modul `src/test/java` im package `de.uniks.pmws2324.ludo`. Implementiere darin einen Test, der ein Spiel mit zwei Spielern startet und diese so gegeneinander spielen lässt, dass es einen Gewinner gibt. Wichtig: Verwende einen **Seed** für die Zufallswerte.

Es dürfen zusätzliche Tests implementiert werden.

[1] Hierbei dürfen keine Methoden gewählt werden, die ausschließlich dazu dienen, weitere Methoden aufzurufen. Getter und Setter sind ebenfalls ausgeschlossen, da sie keine Spiellogik beinhalten.

Informationen zur Abgabe und Präsentation

- Dein Projekt muss pünktlich abgegeben werden (siehe Deadline).
- Im Laufe der zweiten Bearbeitungswoche wird ein Link zur Terminauswahl gepostet (Ankündigung über Discord), über den sich die Studierenden einen Zeitslot auswählen müssen, in dem sie ihr Projekt präsentieren. Diese Präsentationen werden erst nach der Abgabe stattfinden.
- Die Präsentation wird in Präsenz in unserem Fachgebiet stattfinden. Du benötigst einen Laptop, um deinen Code und die Anwendung zu zeigen.
- Es soll **keine** PowerPoint-Präsentation vorbereitet werden. Die Ergebnisse des Projektes werden am laufenden Programm, am Programmcode und anderen Projektdateien erklärt.
- Innerhalb von ca. 5 Minuten sollen die Ergebnisse von Aufgabe 2 als Demo präsentiert werden.
- Innerhalb von ca. 10 Minuten soll der Programmcode präsentiert werden.
- Die Prüfenden können jederzeit Fragen zum Code stellen. Nach der Präsentation folgen ein paar Fragen zum Vorlesungsinhalt.
- Die Auswertung wird einige Zeit dauern, daher findet die Notenvergabe erst entsprechend später statt.