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/ws2122/programming-and-modelling/` zu berücksichtigen. Diese Hausaufgabe enthält die Aufgaben des Projekts.

Abgabefrist ist der 03.03.2022 - 23:59 Uhr

Abgabe

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

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

Vorbereitung

Mach dich mit dem Spiel „Vier gewinnt" („Connect Four") vertraut. Es gelten folgende Eigenschaften[1]:

> Connect Four is a two-player connection board game, in which the players choose a color and then take turns dropping colored discs into a seven-column, six-row vertically suspended grid. The pieces fall straight down, occupying the lowest available space within the column. The objective of the game is to be the first to form a horizontal, vertical, or diagonal line of four of one's own discs.

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- bzw. master-Branch gepusht worden sein.

[1] Connect Four - Wikipedia: `https://en.wikipedia.org/wiki/Connect_Four`

Aufgabe 1 - Modellierung

Im Laufe des Projekts soll das Spiel „Vier gewinnt" modelliert und programmiert werden, sodass als Endprodukt ein Programm ensteht, das es zwei Spielern ermöglicht, gegeneinander „Vier gewinnt" 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. Du kannst dich hierfür an der Modellierung des Spielfelds von Nine men's morris orientieren.

Aufgabe 1.1 - Szenarien

Schreibe drei Szenarien zu „Vier gewinnt". 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 - Klassendiagramm mit Fulib umsetzen

Um das Klassendiagramm mit Fulib umsetzen zu können, soll zunächst das Gradle-Projekt initialisiert werden. Hierfür wird, wie aus vergangenen Hausaufgaben bereits bekannt, fulib.org verwendet. Anschließend kann das Datenmodell mit Fulib generiert werden.

Projekt initialisieren

Folgende Schritte müssen auf fulib.org[2] durchgeführt werden:

1. Ändere den gesamten Inhalt der „Scenario"-Textbox zu `# Connect Four` und füge dahinter einen Zeilenumbruch ein.
2. Öffne über den Button „Configure or Export" den Einstellungsdialog.

[2] `https://fulib.org`

3. Gib als package `de.uniks.pmws2122.connectfour.model` und als Projektnamen `PMWS2122_ConnectFour_<GitHubName>` ein.
4. Wähle unter „Format" aus, dass du ein Gradle-Projekt herunterladen möchtest.
5. Lade dein Projekt über den „Export"-Button herunter.
6. Entpacke dein Projekt in dein Repository. Es darf nicht in einem Unterordner in deinem Repository liegen!
7. Öffne dein Projekt mit VSCode und führe `gradle check` aus.
8. Erstelle einen Commit und stelle dabei sicher, dass die `.gitignore` sich in den hinzugefügten Dateien befindet!

Datenmodell generieren

In deinem Projekt gibt es die Klasse GenModel, nutze diese um dein Klassendiagramm aus Aufgabe 1.3 zu generieren.

Aufgabe 1.5 - Wireframes

Erstelle Wireframes für einen SetupScreen und einen IngameScreen deines Spiels. Lege die pdf-Dateien wie folgt in deinem Repository ab:

`modelling/wireframes/<Setup bzw. Ingame>Screen.pdf`

Aufgabe 2 - Programmierung

In dieser Aufgabe wird das „Vier gewinnt"-Spiel vervollständigt. Dazu wird die Oberfläche erstellt und die Spiellogik implementiert. Entsprechend des MVC-Patterns sollen sie durch Controller miteinander verknüpft werden. Außerdem soll durch Tests sichergestellt werden, dass dein Programm funktioniert.

Die Anwendung muss unter Verwendung von JavaFX umgesetzt werden.

Passe deine `build.gradle`-Datei an, um benötigte Dependencies verwenden zu können.

Aufgabe 2.1 - FXML-Dateien

Erstelle anhand der in Aufgabe 1 erstellten Wireframes die entsprechenden FXML-Dateien mit dem Scenebuilder. Lege diese im Modul `src/main/resources` in folgendem Package ab:

`de.uniks.pmws2122.connectfour`

Aufgabe 2.2 - MVC

Implementiere folgende Klassen:

- `de.uniks.pmws2122.connectfour.model.ModelService`
  enthält benötigte Logik-Methoden für das Spiel
- `de.uniks.pmws2122.connectfour.Launcher`
  zum Starten der Anwendung
- `de.uniks.pmws2122.connectfour.StageManager`
  zum Laden der fxml-Dateien und verknüpfen der Controller
- `de.uniks.pmws2122.connectfour.Constants`
  als zentraler Ort für Konstanten
- `de.uniks.pmws2122.connectfour.controller.SetupScreenController`
  als Controller für den SetupScreen
- `de.uniks.pmws2122.connectfour.controller.IngameScreenController`
  als Controller für den IngameScreen
- `de.uniks.pmws2122.connectfour.controller.<THING>SubController`
  als benötigte(r) Sub-Controller

Aufgabe 2.3 - Tests

Lege für drei selbstgewählte Logik-Methoden [3] des ModelServices 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.pmws2122.connectfour.model` ab.

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

Implementiere außerdem einen GUI-Test. Lege hierfür die Klasse `FullGameTest` im Modul `src/test/java` im package `de.uniks.pmws2122.connectfour` an. Implementiere darin einen Test, der ein Spiel mit zwei Spielern startet und diese so gegeneinander spielen lässt, dass es einen Gewinner (ohne Aufgeben) gibt.

Es dürfen zusätzliche Tests implementiert werden.

Informationen zur Präsentation

- Dein Projekt muss pünktlich abgegeben werden (siehe Deadline).
- Im Laufe der zweiten Bearbeitungswoche wird ein Doodle gepostet (Ankündigung über Discord), über das sich die Studierenden einen Zeitslot auswählen müssen, in dem sie ihr Projekt präsentieren. Diese Präsentationen werden erst nach der Abgabe stattfinden.
- Die Präsentation wird über Discord stattfinden. Hierfür ist eine Webcam erforderlich, da wir deine Identität prüfen müssen. Halte hierfür deinen Personalausweis und deinen Studierendenausweis bereit.
- 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 dabei stellen.
- Die Auswertung wird einige Zeit dauern, daher findet die Notenvergabe erst entsprechend später statt.