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 02.02.2024 - 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/Lkz2aske

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 „Vier gewinnt" („Connect Four") vertraut. Es gelten folgende Eigenschaften[1]:

> Das Spiel wird auf einem senkrecht stehenden hohlen Spielbrett gespielt, in das die Spieler abwechselnd ihre Spielsteine fallen lassen. Das Spielbrett besteht aus sieben Spalten (senkrecht) und sechs Reihen (waagerecht). Jeder Spieler besitzt 21 gleichfarbige Spielsteine. Wenn ein Spieler einen Spielstein in eine Spalte fallen lässt, besetzt dieser den untersten freien Platz der Spalte. Gewinner ist der Spieler, der es als erster schafft, vier oder mehr seiner Spielsteine waagerecht, senkrecht oder diagonal in eine Linie zu bringen. Das Spiel endet unentschieden, wenn das Spielbrett komplett gefüllt ist, ohne dass ein Spieler eine Viererlinie gebildet hat.

Auf den folgenden Seiten werden die zu bearbeitenden Aufgaben beschrieben. Das Programm darf insofern frei gestaltet, d.h. mit zusätzlicher Funktionalität und Tests versehen 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.

[1] Vier gewinnt - Wikipedia: https://de.wikipedia.org/wiki/Vier_gewinnt

Aufgabe 1 – Modellierung

Im Laufe des Projekts soll das Spiel „Vier gewinnt" modelliert und programmiert werden, sodass als Endprodukt ein Programm entsteht, 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. Jede pdf-Datei soll einseitig sein.

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

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 – 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 „Vier gewinnt"-Spiel implementiert. 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/cf/view ab.

Aufgabe 2.3 – MVC

Implementiere folgende Klassen:

- de.uniks.pmws2324.cf.Main und de.uniks.pmws2324.cf.App
  zum Starten der Anwendung und Verwalten von Controllern
- de.uniks.pmws2324.cf.service.GameService
  enthält benötigte Logik-Methoden für das Spiel
- de.uniks.pmws2324.cf.Constants
  als zentraler Ort für Konstanten
- de.uniks.pmws2324.cf.controller.SetupController
  als Controller für den Setup-Bildschirm
- de.uniks.pmws2324.cf.controller.IngameController
  als Controller für den Ingame-Bildschirm
- de.uniks.pmws2324.cf.controller.GameOverController
  als Controller für den GameOver-Bildschirm
- de.uniks.pmws2324.cf.controller.<THING>Controller
  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[2] des `GameServices` je eine eigene Test-Methode im `GameServiceTest` an. Verwende hierfür die bereits vorhandene Klasse `GameServiceTest` im Modul `src/test/java` im package de.uniks.pmws2324.cf.service. Überprüfe die Funktionalität der Logik-Methode in den entsprechenden Test-Methoden durch sinnvolle und zielgerichtete Tests. Erkläre in jeder Methode in Form von Kommentaren, welche Anforderung an das Verhalten der Logik-Methode geprüft wird.

Implementiere außerdem einen GUI-Test. Verwende hierfür die Klasse `FullGameTest` im Modul `src/test/java` im Package de.uniks.pmws2324.cf. 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.

[2] 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.