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.

Abgabefrist ist der 20.01.2022 - 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 benötigst du **kein neues** Repository. Es wird das gleiche Repository benutzt, das bereits in Hausaufgabe 4 angelegt wurde. Dieses kann über folgenden Link eingesehen oder auch erstellt werden, falls nicht bereits geschehen:

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

Nicht oder zu spät gepushte (Teil-)Abgaben werden mit 0 Punkten bewertet.

Abgaben, die nicht lauffähig sind, werden mit 0 Punkten bewertet!

Das Ignorieren der Commit Message-Vorgaben wird mit 0 Punkten bewertet!

Projekte, deren GUI nicht mit FXML-Dateien umgesetzt sind, werden mit 0 Punkten bewertet!

Alle Tests (alt/neu) müssen nach wie vor funktionieren, sollte dies nicht der Fall sein, wird mit 0 Punkten bewertet!

Vorbereitungen

Musterlösung der vorherigen Aufgaben

Auch für diese Aufgabe kannst du auf deinen bestehenden Code aufbauen oder unsere Musterlösung[1] verwenden.

Gradle anpassen

Solltest du nicht die Musterlösung verwenden, ergänze bitte deine `build.gradle` wie hier[2] gezeigt.

[1] `https://github.com/sekassel/pmws2122-files/tree/main/HA07`
[2] `https://github.com/sekassel/pmws2122-files/blob/main/HA07/build.gradle`

Aufgabe 1 - ModelService (30P)

In dieser Aufgabe soll der `ModelService` erweitert werden, sodass alle nötigen Logikfunktionen für das Spiel fertigstellt sind. Hierfür müssen folgende Schritte ausgeführt werden, um bereits zuvor geschriebene Funktionalität zu erweitern.

Aufgabe 1.1 Vorgegebene Klassen

Wie in der letzten Hausaufgabe, geben wir auch hier den `ModelService` vor, damit wir einen einheitlichen Startpunkt haben. Du kannst natürlich auch deinen eigenen Code weiterverwenden, vergleiche daher bitte die Struktur deiner Klasse mit unserer Vorlage. Achte darauf, dass der `ModelService` im folgenden Package liegt

`de.uniks.pmws2122.model.ModelService`[3]

im Ordner src/main/java.

Aufgabe 1.2 Funktionalität

Während der Entwurfsphase eines Algorithmus können mithilfe von Kommentaren einzelne Schritte einer Methode skizziert werden. Als Gedankenstütze ist es üblich, solche Kommentare in folgender Form zu schreiben

```
// TODO: Add player to the board
```

Die meisten Entwicklungsumgebungen[4] markieren das `TODO:` farblich, wodurch leicht erkennbar ist, an welchen Stellen eine Methode unvollständig ist. Daher wurden in der Codevorlage des `ModelService` entsprechend zu ergänzende Methoden und Codeteile durch ein `TODO:` markiert.

Folgende Methoden müssen erweitert oder vollständig implementiert werden

```
checkWinner()
nextTurn()
checkNextPhase()
placeMan(Field field)
moveMan()[5]
removeMan(Man man)
checkVerticalMill(Man lastPutMan)
```

orientiere dich dabei an den markierten Kommentaren. Entferne den Zusatz `TODO:` aus den Kommentaren, sofern du die geforderte Funktionalität implementiert hast.

Nicht entfernte `TODO:` führen zu Minuspunkten.

[3] https://github.com/sekassel/pmws2122-files/blob/main/HA07/ModelService.java
[4] Sollte VSCode dies bei dir nicht tun, installiere die Erweiterung *TODO Highlight*
[5] Beachte die vorgegebenen Variablen im *ModelService*

Aufgabe 2 - Controller (10P)

In dieser Aufgabe muss der `IngameScreenController` angepasst und ein neuer `FieldSubController` erstellt werden. Dafür müssen folgende Schritte durchgeführt werden:

Aufgabe 2.1 Vorgegebene Klassen

Vergleiche deine Implementierung des `IngameScreenController` mit der Musterlösung und verwende entweder diese oder deinen eigenen Code. Den `FieldSubController` musst du aus der Vorlage übernehmen. Achte darauf, dass die Dateien in den richtigen Packages liegen

`de.uniks.pmws2122.controller.IngameScreenController`[6]
`de.uniks.pmws2122.controller.FieldSubController`[7]

im Ordner src/main/java.

Aufgabe 2.2 Funktionalität

Auch in dieser Aufgabe sind die relevanten zu ergänzenden Codestücke mit einem `TODO:` im Kommentar versehen.

Im `IngameScreenController` muss für jedes Feld, welches mit dem Spiel verbunden ist, ein `FieldSubController` erstellt und initialisiert werden. Diese müssen in der `stop()` Methode wiederum gestoppt und gelöscht werden. Nutze die dafür angelegte Variable `fieldCons`. Abschließend soll dem Nutzer eine Auswahlmöglichkeit gezeigt werden, bei der er wählen kann welcher Spieler beginnt.

Im `FieldSubController` müssen die gegebenen `ActionListener` mit der `view` verbunden werden. Anschließend soll ein Effekt implementiert werden, durch den ein Feld seine Farbe ändert, wenn über dieses mit der Maus gefahren wird. Verwende dafür die Methode `setFill(...)` auf der `view`.

Entferne den Zusatz `TODO:` aus den Kommentaren, sofern du die geforderte Funktionalität implementiert hast.

Nicht entfernte `TODO:` führen zu Minuspunkten.

[6] https://github.com/sekassel/pmws2122-files/blob/main/HA07/IngameScreenController.java
[7] https://github.com/sekassel/pmws2122-files/blob/main/HA07/FieldSubController.java

Aufgabe 3 - Testen (14P)

In dieser Aufgabe sollen die Erweiterungen des `ModelService` getestet werden.

Erstelle bzw. erweitere unter src/test/java im Package de.uniks.pmws2122.model folgende Testklassen mit entsprechenden Methoden:

Test-Klasse `VerticalMillTest` mit den Methoden
`testMillNoMill()`
`testMillOldMill()`
`testMillNewMillVerticalFromMid()`
`testMillNewMillVerticalFromTop()`
`testMillNewMillVerticalFromBottom()`

Test-Klasse `WinnerTest` erweitern um die Methode
`public void testWinNoMovingPossible()`

Test-Klasse `ManTest` mit den Methoden
`testManPlacedEmptyField()`
`testManMoveEmptyField()`
`testManRemove()`
`testManPlacedOccupiedField()`
`testManMoveOccupiedField()`
`testManRemoveNullMan()`

Die Tests sind, wie aus den vorherigen Hausaufgaben bekannt, zu implementieren. Beachte, dass alle vorher angelegten Tests ebenfalls positiv durchlaufen müssen! Passe nötigenfalls alte Test-Klassen an, damit dies der Fall ist.

Sollten die Tests nicht lauffähig sein oder nicht positiv durchlaufen, führt dies zu 0 Punkten!