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 27.01.2022 - 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.

Vergleiche deine `moveMan()` Methode aus dem `ModelService` mit der Musterlösung, hier wurde ein Fehler aus der letzten Hausaufgabe behoben.

[1] `https://github.com/sekassel/pmws2122-files/tree/main/HA08`

Aufgabe 1 - Property Change Listener (10P)

Ziel dieser Aufgabe ist es, Änderungen des Datenmodells während der Laufzeit in der Oberfläche anzuzeigen. Dafür werden PropertyChangeListener verwendet, welche an Modellobjekten angemeldet werden können.

1.1 IngameScreenController

Vergleiche deine Abgabe mit der Musterlösung [2], verwende diese oder ergänze deine Lösung mit den vorgegebenen `TODO:` Kommentaren. Für diese Aufgaben müssen weitere kleinere Anpassungen am `IngameScreenController` vorgenommen werden.

Zusätzlich zu den bereits initialisierten Oberflächen-Elementen, müssen die Elemente `currentPlayerColorDisplay` und `currentPlayerActionLabel`, wie vorher gelernt, geladen werden.

Anschließend müssen die `PropertyChangeListener`, wie in der Übung gezeigt, an- bzw. abgemeldet werden. Die entsprechenden Stellen wurden durch Kommentare in der `init()`- bzw. `stop()`-Methode markiert.

Im nächsten Schritt sollte die Methode `onGivenUpButtonPressed` angepasst werden. Beachte den Kommentar.

Abschließend musst du die drei `PropertyChangeListener`-Methoden `onWinnerChanged`, `onCurrentPlayerChanged` und `onCurrentPlayerActionChanged` implementieren. Orientiere dich an den Kommentaren.

1.2 FieldSubController

Vergleiche deine Abgabe mit der Musterlösung [3], verwende diese oder ergänze deine Lösung mit den vorgegebenen `TODO:` Kommentaren.

Als erstes müssen die `PropertyChangeListener`, wie in der Übung gezeigt, an- bzw. abgemeldet werden. Die entsprechenden Stellen wurden durch Kommentare in der `init()`- bzw. `stop()`-Methode markiert.

Abschließend musst du die `PropertyChangeListener`-Methode `onManPlacedChanged` implementieren. Orientiere dich an den Kommentaren.

Wichtige Hinweise zur Implementierung:

- PropertyChangeListener müssen korrekt abgebaut werden, wenn die stop()-Methode aufgerufen wird.
- Eventuell müssen PropertyChangeListener nicht nur in init() und stop(), sondern auch bei Änderung von Objekten an- und abgemeldet werden.

Nicht entfernte `TODO:` führen zu Minuspunkten.

[2] `https://github.com/sekassel/pmws2122-files/blob/main/HA08/IngameScreenController.java`
[3] `https://github.com/sekassel/pmws2122-files/blob/main/HA08/FieldSubController.java`

Aufgabe 2 - Spiel fertigstellen (31P)

In dieser Aufgabe musst du die letzten Funktionalitäten für das Spiel implementieren.

2.1 IngameScreenController

Passe, falls noch nicht geschehen, den `chooseChoiceDialog` in der `init()`-Methode wie im Kommentar gefordert an.

2.1 FieldSubController

Abschließend muss das Verhalten beim Klick auf ein Feld implementiert werden. Die grundlegende Struktur dafür wurde bereits in der Vorlage[4] implementiert. Es gilt, die Methoden `gamePhasePlacingManNull`, `gamePhasePlacingManNotNull`, `gamePhaseMovingManNull` und `gamePhaseMovingManNotNNull` zu implementieren.

Vergleiche deine Lösung und ergänze die Methoden. Aufgrund der Komplexität wird in den zu implementierenden Methoden die grobe Struktur vorgegeben. Die Kommentare markieren die Stellen, an denen du Code ergänzen musst.

Nicht entfernte `TODO:` führen zu Minuspunkten.

[4] `https://github.com/sekassel/pmws2122-files/blob/main/HA08/FieldSubController.java`

Aufgabe 3 - Final Testination (10P)

Zu guter Letzt muss der kritische Pfad der Anwendung getestet werden. Das bedeutet, ein Test muss die nötigen Aktionen durchführen, damit ein Spieler das Spiel gewinnt.

Erstelle dafür unter src/test/java im Package de.uniks.pmws2122 die Testklasse `FullGameTest`. Implementiere mithilfe von TestFX einen Test, welcher einen beliebigen Spieler gewinnen lässt. Achte dabei darauf, an geeigneten Stellen asserts zu verwenden, um den Status der Anwendung zu überprüfen.

Abschluss:

Nach Implementierung der Funktionalität dieser Hausaufgabe ist die Anwendung NineMenMorris für dieses Semester abgeschlossen. Einige wenige fehlende Features dürfen bei gegebener intrinsischer Motivation umgesetzt werden, sind jedoch nicht Pflicht. In der kommenden Hausaufgabe für Studierende der Informatik wird eine neue Anwendung begonnen.

Für die Studierenden der Mechatronik ist diese (Hausaufgabe 8) die letzte reguläre Hausaufgabe. In der kommenden Woche wird das Aufgabenblatt zum Projekt für Studierende der Mechatronik veröffentlicht.