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 16.12.2021 - 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 benötigt ihr **kein neues** Repository. Es wird das gleiche Repository benutzt, das bereits in Hausaufgabe 4 angelegt wurde. Dieses kann über folgenden Link angelegt 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!**

## Vorbereitungen

### Anpassungen der build.gradle

In dieser Hausaufgabe wird das Oberflächenframework JavaFX, sowie die TestFX-Library verwendet. Hierzu soll die bestehende build.gradle erweitert werden, um deinem Projekt die genannten Frameworks hinzuzufügen. Die hinzuzufügenden Zeilen sind in der zur Verfügung gestellten build.gradle-Datei[1] zu finden und müssen an die entsprechenden Stellen der build.gradle-Datei deines eigenen Projektes kopiert werden.

[1] Datei: `https://github.com/sekassel/pmws2122-files/blob/main/HA06/build.gradle`

# Aufgabe 1 - Fxml (23P)

In dieser Aufgabe soll die Grundlage zur Umsetzung der in der vorherigen Hausaufgabe erstellten Wireframes geschaffen werden.

Wir verwenden für die Erstellung unserer Oberflächen den SceneBuilder. Lade dir den SceneBuilder unter folgendem Link herunter:

`https://gluonhq.com/products/scene-builder/`

Der Scenebuilder speichert seine Dateien im fxml-Format. Für jedes Wireframe muss eine eigene fxml-Datei erstellt werden. Die zwei erstellten fxml-Dateien sind im Modul src/main/resources im Package de.uniks.pmws2122 abzulegen. Achte darauf, dass im resources-Ordner die korrekt geschachtelte Ordnerstruktur des erforderten Packages erstellt wurde und nicht nur ein Ordner mit dem Namen "de.uniks.pmws2122" existiert.

Wenn du deine eigenen Wireframes nicht umsetzen möchtest, können die von uns zur Verfügung gestellten Wireframes (Abbildung 1 und 2) als Arbeitsgrundlage verwendet werden.

Sowohl beim Umsetzen deiner eigenen als auch unserer Wireframes sind vorgegebene Benennungen der fx:id bestimmter Komponenten zu berücksichtigen.

Im SetupScreen werden folgende fx:id vergeben:

- `playerNameBlackInput` für die Text-Eingabe für den Spieler mit Farbe Schwarz
- `playerNameWhiteInput` für die Text-Eingabe für den Spieler mit Farbe Weiß
- `createGameButton` für den Create Game-Button

Im IngameScreen werden folgende fx:id vergeben:

- `currentPlayerNameLabel` für das Label zum Anzeigen des Namens des aktuellen Spielers
- `currentPlayerColorDisplay` für die Komponente (z. B. Circle), die die Farbe des aktuellen Spielers anzeigt
- `currentPlayerActionLabel` für das Label zum Anzeigen der Aktion des aktuellen Spielers
- `giveUpButton` für den Give up-Button
- `<kleiner Buchstabe><Zahl>FieldDisplay` für die Komponente (z. B. Circle), die das entsprechende Feld repräsentiert (z. B. `e5FieldDisplay`, insgesamt 24 Stück, Anordnung entsprechend des bekannten Spielfelds[2])

[2]Spielfeld: `https://en.wikipedia.org/wiki/Nine_men%27s_morris#/media/File:Nine_Men's_Morris_board_with_coordinates.svg`

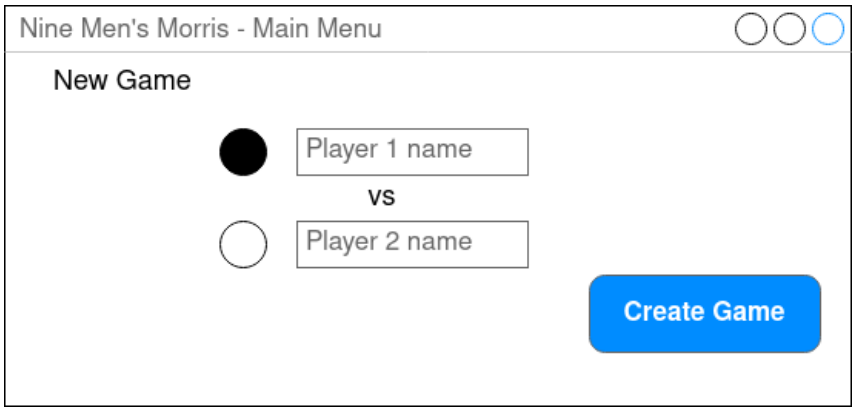


Abbildung 1: SetupScreen

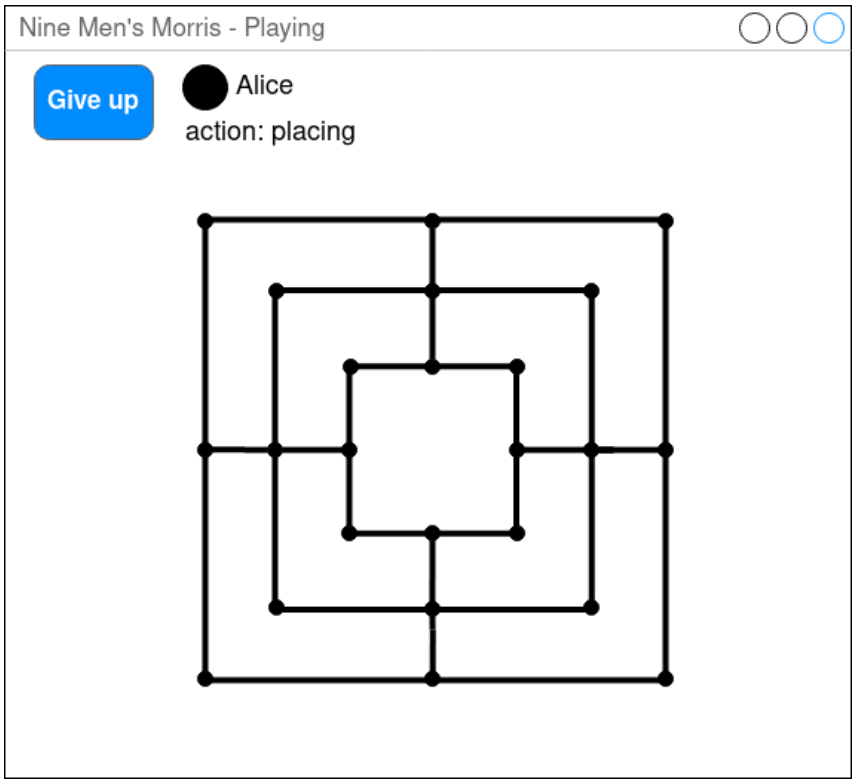


Abbildung 2: IngameScreen

Committe und pushe die neuen Dateien abschließend auf den main-Branch.

**Achte darauf, das Repository der aktuellen Hausaufgabe zu verwenden.**

# Aufgabe 2 - JavaFX (22P)

## Aufgabe 2.1 Vorgegebene Klassen

Um einen einheitlichen Startpunkt zur Implementierung der JavaFx-Applikation zu schaffen, wurden folgende vier Java-Klassen bereitgestellt.

`de.uniks.pmws2122.Launcher` [3]
`de.uniks.pmws2122.StageManager` [4]
`de.uniks.pmws2122.controller.SetupScreenController` [5]
`de.uniks.pmws2122.controller.IngameScreenController` [6]

Füge diese in das entsprechende Package unter src/main/java in dein Projekt ein.

Des Weiteren wurde die Constants.java-Datei[7] um zwei neue String-Konstanten erweitert, die du bei dir ebenfalls ergänzen musst.

## Aufgabe 2.2 Funktionalität

In dieser Aufgabe soll eine Verbindung zwischen fxml-Dateien und Controller geschaffen werden. Implementiere hierzu die Methoden der kopierten Klassen. Durch Kommentare in den Methoden wird deren Funktionalität bereits vorgegeben. Nach Bearbeitung der Aufgabe sollte es möglich sein, durch das Klicken des Create Game-Buttons vom SetupScreen in den IngameScreen zu gelangen. Ebenso sollte dies umgekehrt durch den Give up-Button möglich sein.

Weiterhin sollen an dieser Stelle die **Fenstertitel** umgesetzt werden, die in den Wireframes der vorherigen Aufgabe zu sehen sind. In Abbildung 1 ist dieser „Nine Men's Morris - Main Menu"und in Abbildung 2 „Nine Men's Morris - Playing". Nutze dafür die neuen String-Konstanten der Constants.java-Datei.

Sollten die Fenstertitel deiner eigenen Wireframes abweichen, setze trotzdem die hier genannten Fenstertitel um. Du kannst die Fenstertitel deiner eigenen Wireframes gerne an die hier genannten anpassen.

Committe und pushe die Änderung abschließend auf den main-Branch.

**Achte darauf, das Repository der aktuellen Hausaufgabe zu verwenden.**

[3] Launcher:
`https://github.com/sekassel/pmws2122-files/blob/main/HA06/Launcher.java`
[4] StageManager:
`https://github.com/sekassel/pmws2122-files/blob/main/HA06/StageManager.java`
[5] SetupScreenController:
`https://github.com/sekassel/pmws2122-files/blob/main/HA06/SetupScreenController.java`
[6] IngameScreenController:
`https://github.com/sekassel/pmws2122-files/blob/main/HA06/IngameScreenController.java`
[7] Constants:
`https://github.com/sekassel/pmws2122-files/blob/main/HA06/Constants.java`

# Aufgabe 3 - TestFX (13P)

Als Abschluss für diese Hausaufgabe sollen die eben implementierten Teile getestet werden. Dies geschieht wie in der Vorlesung gezeigt mithilfe der hinzugefügten TestFx-Library.

## Aufgabe 3.1 Vorgegebene Klasse

Nutze die zur Verfügung gestellte Test-Klasse, um mit der Implementierung des TestFx-Testes zu beginnen.

`de.uniks.pmws2122.StageManagerTest` [8]

Füge den StageManagerTest unter src/test/java in das vorgegebene Package in dein Projekt ein.

## Aufgabe 3.2 Implementierung

Implementiere die start-Methode wie in der Übung gezeigt.

Die changeViewTest-Methode soll folgenden Ablauf implementieren:

- Initialen Fenstertitel prüfen
- Spielernamen in die dafür vorgesehenen Eingabefelder eingeben. Verwende für den schwarzen Spieler Bob und für den weißen Spieler Caro
- Create Game-Button klicken, um ein neues Spiel mit den Spielern zu starten
- Neuen Fenstertitel prüfen
- Prüfen, ob der Name des aktuellen Spielers im entsprechenden Label eingetragen ist
- Give up-Button klicken
- Fenstertitel erneut prüfen
- Prüfen, dass die Eingabefelder für die Spielernamen leer sind

Committe und pushe die Änderung abschließend auf den main-Branch.

**Achte darauf, das Repository der aktuellen Hausaufgabe zu verwenden.**

[8] StageManagerTest:
`https://github.com/sekassel/pmws2122-files/blob/main/HA06/StageManagerTest.java`

# Anhang

Es folgt eine Auflistung hilfreicher Webseiten und weiterer Erklärungen zu den Themen dieser Hausaufgabe. Die Links sind als Startpunkt zur selbstständigen Recherche angedacht. Das Durcharbeiten der folgenden Quellen ist kein bewerteter Anteil der Hausaufgaben.

## Zur Verfügung gestellte Dateien

- `https://github.com/sekassel/pmws2122-files/tree/main/HA06`

## Scene Builder

- Scene Builder Download: `https://gluonhq.com/products/scene-builder/`

## TestFX

- TestFX auf GitHub: `https://github.com/testfx/testfx`