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/ws1920/programming-methodologies/` zu berücksichtigen.

Abgabefrist ist der 05.12.2019 - 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 Korrekteure sichtbar. Für diese und voraussichtlich alle zukünftigen Abgaben wird kein neues Repository benötigt. Nutzt weiterhin jenes, welches mit folgendem Link angelegt wurde:

`https://classroom.github.com/a/X9QUkSjx`

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

Vorbereitung

Zur Bearbeitung der Hausaufgabe sollte eine Entwicklungsumgebung verwendet werden. Wir empfehlen aufgrund der Nachvollziehbarkeit die Verwendung von IntelliJ (siehe Aufgabenblatt 3). Die Abgabe muss als **lauffähiges** Projekt abgegeben werden.

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

Commit Message Konventionen

In dieser und zukünftigen Hausaufgaben führen wir eine Konvention für die Commit Messages ein.

Solche Commits, welche als Abgabe einer Hausaufgabe dienen, werden mit der Nachricht: ”Final HA<Number>” versehen. Für die Abgabe von Teilaufgaben (wie in Aufgabe 1 dieses Blattes), wird folgende Konvention eingeführt ”HA<Number> finished Task<TaskNumber>”.

Falls in Folge der Abgabe-Commits ein Fehler auffällt, welchen ihr zeitlich gesehen noch korrigieren könnt, so verwendet für den Folge-Commit wieder die Message ”Final HA<Number>”. Der aktuellste Commit mit dieser Nachricht, vor der Deadline, wird als Abgabe gewertet.

Das Ignorieren der Konventionen wird mit 0 Punkten bewertet!

Aufgabe 1 - Mockups (50P)

In dieser Aufgabe soll mithilfe der Webseite Fulib.org eine Einführung in die Erstellung von Mockups geschehen. Dabei werden diese Mockups mit der Webseite selbst erstellt. Die Abbildungen 1 bis 3 zeigen je eine Skizze (Design wird abweichen) der zu erstellenden Mockups.

Erstellt mithilfe von Fulib.org 3 verschiedene Mockups, die den unten aufgeführten gleichen, und fasst eure Szenarios in einem Clickdummy zusammen. (Abbildung 4 enthält eine Zusammenfassung der Regeln/Sätze)

Legt den Text dann unter src/main/scenarios/de/uniks/pmws1920/mockups als mockupScenario.md ab und lasst alle Dateien mit dem gradle check-Task generieren.

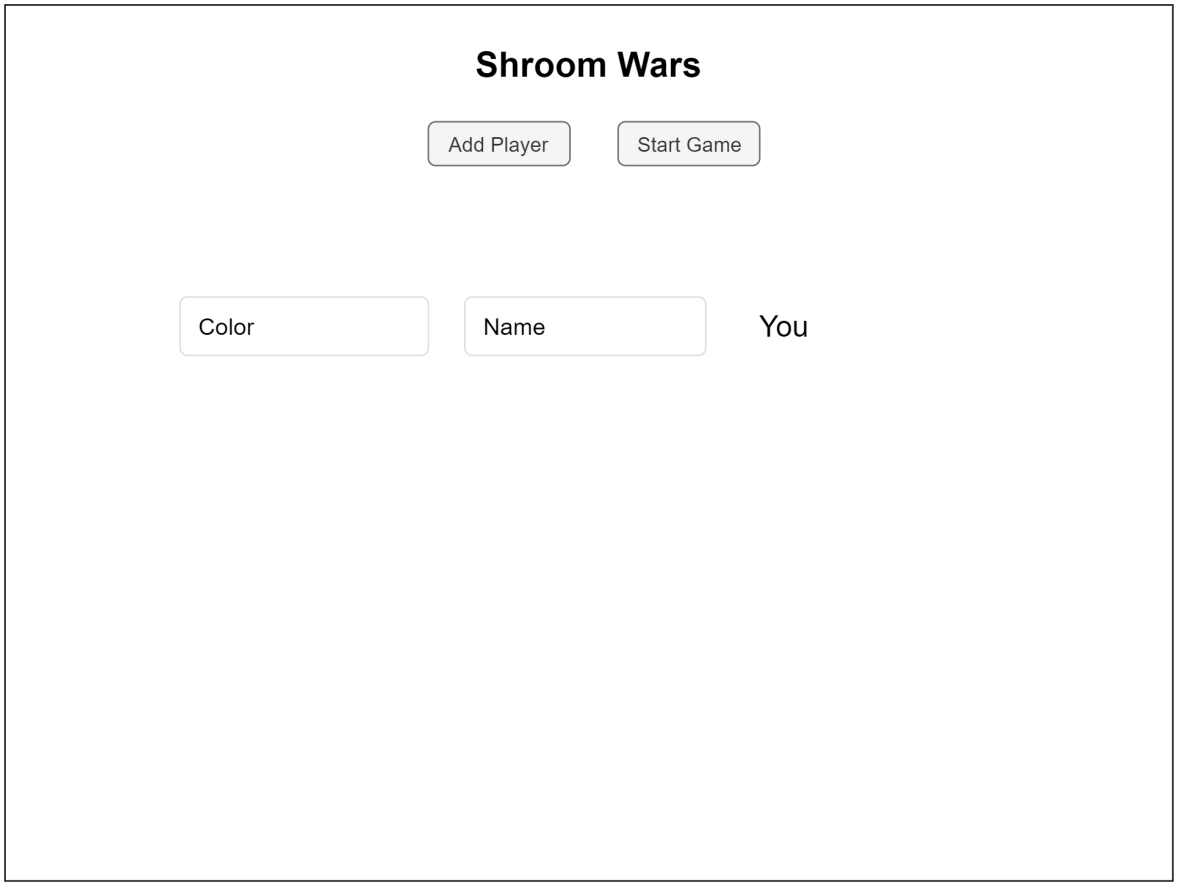

Abbildung 1: Mockup erste HTML

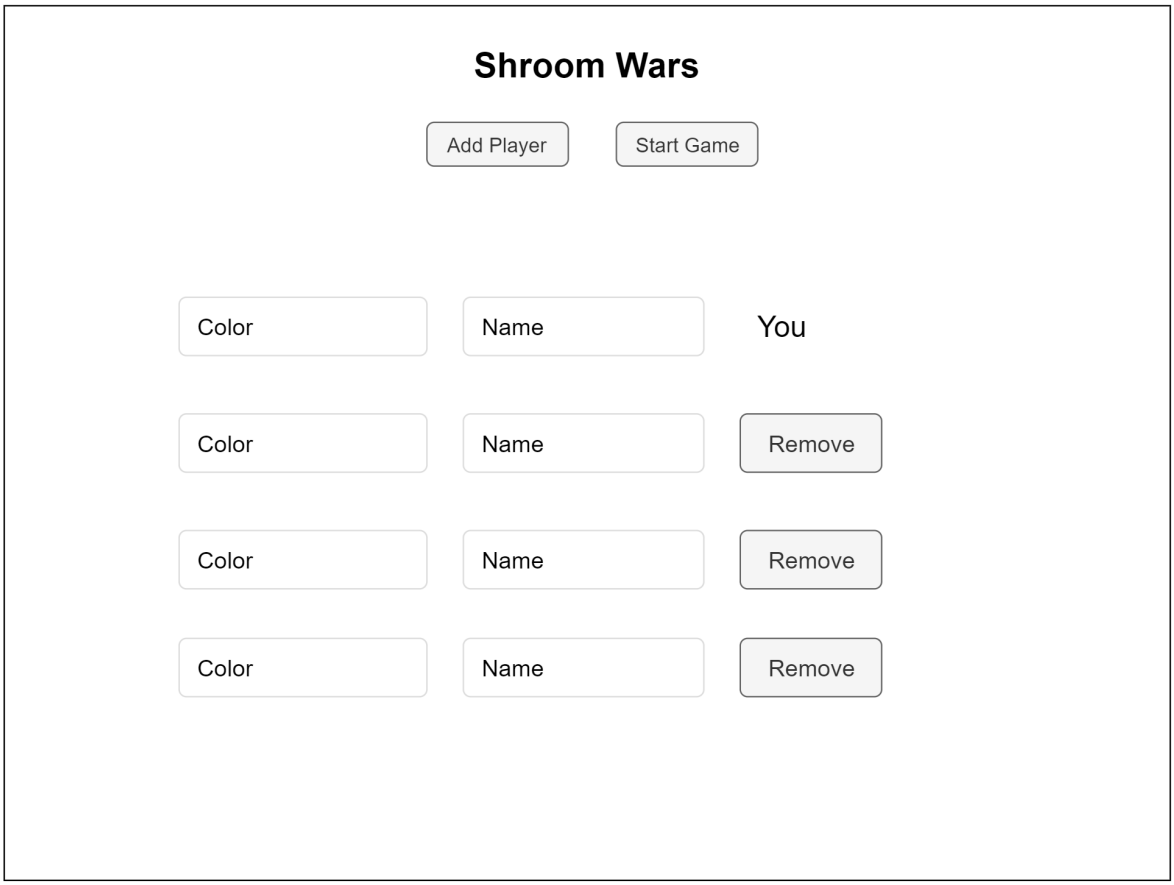

Abbildung 2: Mockup zweite HTML

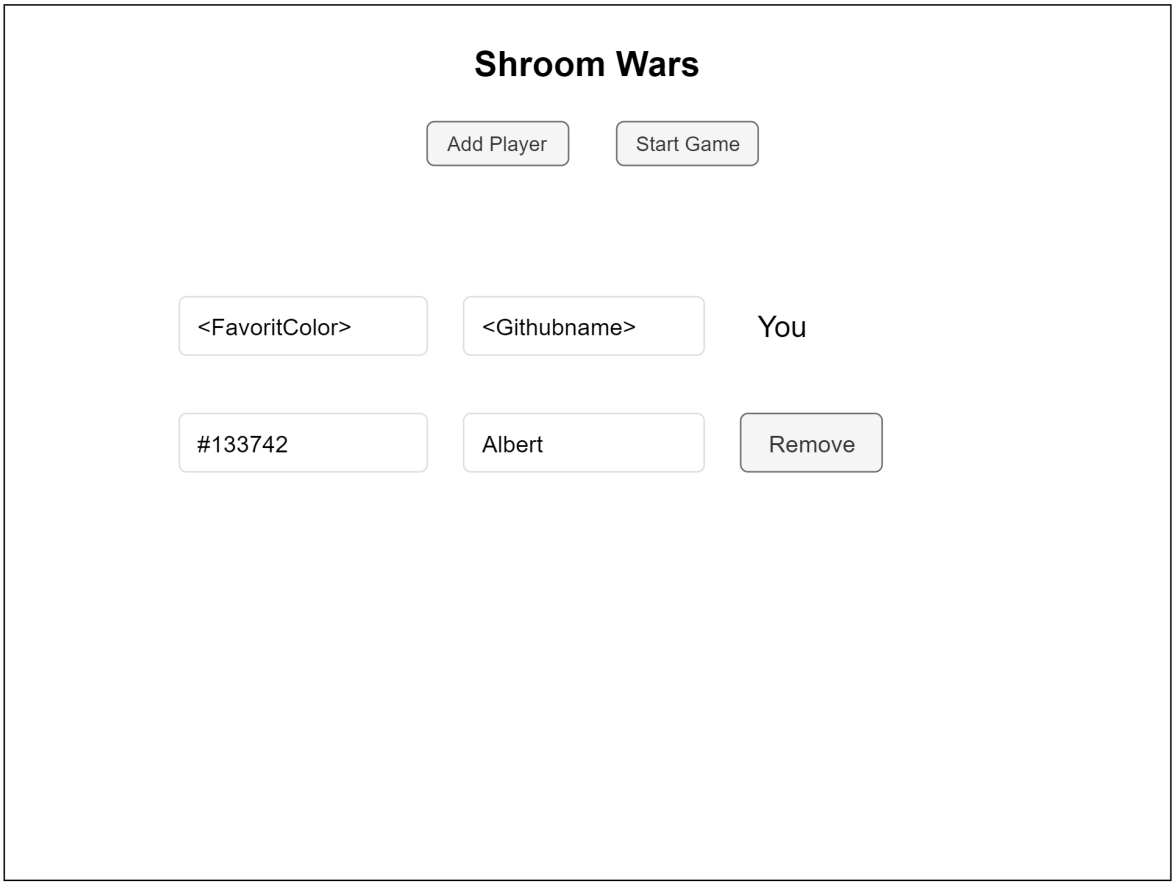

Abbildung 3: Mockup dritte HTML

Committe und pushe die Änderung an deinem Gradle-Projekt abschließend auf den master-Branch.

Bei der Bewertung wird vor allem auf die korrekte Umsetzung der Mockups geachtet.

Markiert den letzten Commit für diese Teilaufgabe der Konvention entsprechend.

```
// Create App with Title
There is an app with id <AppID> and with description "<AppTitle>".

// Create page
There is a page with id <PageID>.

// Create a line container with content
There is a line with id <LineId> and with (description "<LineRegEx>" || <CellID> ).

// Create a cell with content
There is a cell with id <CellID> and with text "<CellRegEx>".



// Set page content to App
<AppID> has content <PageID>.

// Set page content
<PageID> has content <LineID> (and <LineID>)*.



// Content manipulation

// Set cell value or text
We write "<SimpleText>" into (text || value) of <CellID>.

// Add lines to page
We add <LineID> (and <LineID>)* to content of <PageId>.

// Remove line from page
We remove <LineID> from content of <PageID>.



// LineRegEx: Regular Expression for a Line
LineRegEx = <CellRegEx> (| <CellRegEx>)*

// CellRegEx: Regular Expression for a Cell
<CellRegEx> = input <InitialInputText> || button <ButtonText> || SimpleText

// Dump Mockup into HTML File
![<AppName>](<Filename>.html)

// Dump Clickdummy into HTML File
![<AppName>](<Filename>.mockup.html)


// Syntax & Semantik
|| : Logisches Oder
| : Trennzeichen innerhalb eines Ausdrucks
```

Abbildung 4: How to Fulib.org Mockup

Abbildung 4 zeigt eine Zusammenfassung aller notwendigen Szenario-Befehle. Diese sind in Anlehnung an reguläre Ausdrücke formuliert.

Aufgabe 2 - JavaFx (50P)

In der folgenden Aufgabe soll das Oberflächenframework JavaFx in das Projekt eingebunden und vorbereitend eingerichtet werden. Hierzu soll die build.gradle-Datei um folgende Zeilen erweitert werden. Nachfolgend sollte ein Gradle refresh durchgeführt werden.

```
plugins {
    ...
    id 'application'
    id 'org.openjfx.javafxplugin' version '0.0.8'
}

javafx {
    version = "12"
    modules = ['javafx.graphics', 'javafx.fxml', 'javafx.base', 'javafx.controls']
}

mainClassName = '<Launcherklasse>' (Mit package angeben und ohne Dateiendung)
(zB. de.uniks.pmws1920.Launcher)
```

Erstellt nun zwei Klassen. Eine mit dem Namen Launcher.java und die andere mit dem Namen SetupScreen.java. Beide sollen im Package de.uniks.pmws1920 abgelegt werden. Lasst nun die Klasse SetupScreen von javafx.application.Application erben. Die folgenden Ausschnitte zeigen ein Beispiel der entstandenen Dateien:

```java
package de.uniks.pmws1920;

public class Launcher {
    public static void main(String[] args) {
        SetupScreen.launch(SetupScreen.class, args);
    }
}
```

```java
package de.uniks.pmws1920;

import javafx.application.Application;
import javafx.stage.Stage;

public class SetupScreen extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception {
        // Put code here
    }
}
```

Füge nun der start()-Methode den aus der Vorlesung bekannten Code hinzu, um ein Fenster zu öffnen. Das Fenster sollte das Format 800(Breite) x 600(Höhe) aufweisen.

Setze in der start()-Methode zusätzlich das in Abbildung 1 gezeigte Mockup in die Tat um. Verwende jedoch für die Farbe des Spielers anstelle eines Textfeldes die JavaFx-Komponente javafx.scene.control.ColorPicker. Für die Textfelder verwende die Komponente javafx.scene.control.TextField.

(Das Aufteilen des Codes mithilfe von selbst geschriebenen Hilfsmethoden ist ausdrücklich erlaubt. Versuche den Code damit ein wenig übersichtlicher zu gestalten)

Committe und pushe die Änderung an deinem Gradle-Projekt abschließend auf den master-Branch.

Bei der Bewertung wird vor allem auf die Lauffähigkeit der Anwendung und die gesamtanzahl der GUI-Elemente geachtet.

Markiert den letzten Commit für diese Hausaufgabe der Konvention entsprechend.

Anhang

Es folgt eine Auflistung hilfreicher Webseiten und weiteren 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.

JavaFx

- Erklärung zur Launcher-Klasse:
  `https://stackoverflow.com/a/52631238/10526222`