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 03.02.2022 - 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 ein **neues Repository**. Dieses kann über folgenden Link erstellt werden:

`https://classroom.github.com/a/xVKNlCc6`

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 müssen funktionieren, sollte dies nicht der Fall sein, wird mit 0 Punkten bewertet!

Aufgabe 1 - Projekt anlegen (11P)

Ziel dieser Hausaufgabe ist es, das bereits in den vergangenen Übungen erlernte Wissen anzuwenden, um die Grundlage einer Server/Client-Applikation zu erstellen. Hierfür wird wie zuvor bereits beschrieben ein neues Repository und somit auch ein neues Projekt angelegt (vgl. Seite 1 **Abgabe**).

Führe die Schritte zum Herunterladen eines Java-Projektes von fulib.org analog zu Hausaufgabe 4 aus, unter Beachtung der nachfolgenden Punkte. Es gelten folgende Vorgaben zu den zu verwendenden Technologien, sowie der Projektstruktur:

- Das Gradle-Projekt muss den Namen PMWS2122_ICP_<Githubname> haben.
- Die .gitignore muss vor dem ersten Commit eingepflegt werden! Die Datei befindet sich im heruntergeladenen Projekt und darf nicht gelöscht werden.
- Für die Oberflächen muss JavaFx verwendet werden.
- Die Anwendung muss mittels MVC-Pattern implementiert werden.
  - Das Modell wird in das Package de.uniks.pmws2122.icp.model generiert.
  - FXML-Dateien werden in src/main/resources im Package de.uniks.pmws2122.icp.view abgelegt.
  - Controller werden im Package de.uniks.pmws2122.icp.controller abgelegt.
- Analog zu den vorherigen Hausaufgaben müssen alle schreibenden Zugriffe auf das Datenmodell über einen `ModelService` geschehen.
  Dieser ist im Package de.uniks.pmws2122.icp.model abzulegen.
- Analog zu den vorherigen Hausaufgaben müssen alle Szenenwechsel über einen StageManager geschehen. Dieser ist im Package de.uniks.pmws2122.icp abzulegen.
- `String`-Konstanten sollen in der Klasse Constants im Package de.uniks.pmws2122.icp abgelegt werden.

Datenmodell

Das folgende Datenmodell ist eine bindende Vorgabe und darf nicht verändert werden. Es muss mithilfe von Fulib und der GenModel-Klasse generiert werden.

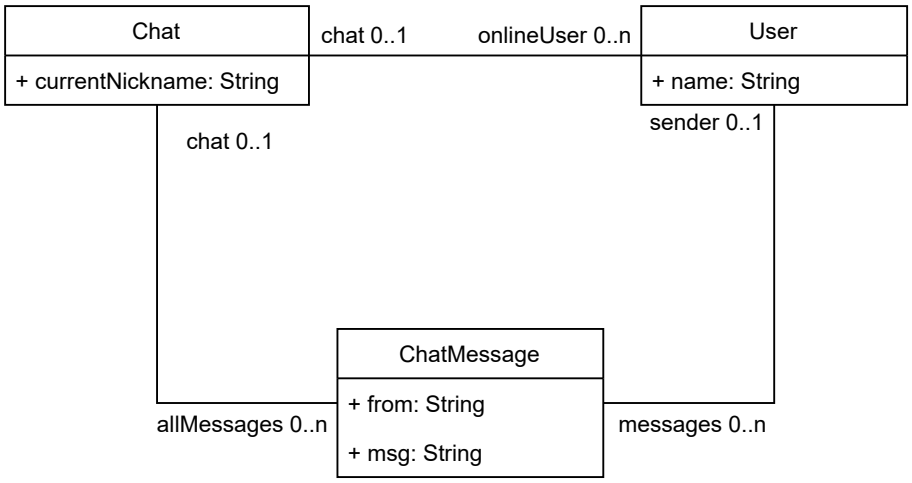

Abbildung 1: Datenmodell ICP

Aufgabe 2 - Oberfläche (8P)

Am Ende dieser Aufgabe sollen mindestens zwei FXML-Dateien zu den folgenden Mockups erstellt werden. Die folgenden Mockups sind lediglich Vorlagen, aber keine Vorgaben. Es ist dir somit freigestellt, das Design zu verändern. Es ist notwendig, dass die beschriebenen GUI-Elemente mit den vorgegebenen fx:ids und die Funktionalität der Screens in deiner Umsetzung enthalten sind. Bis auf den Szenenwechsel wird diese Funktionalität erst in den kommenden Hausaufgaben implementiert.

Aufgabe 2.1 - LoginScreen

Der LoginScreen bildet den Eintrittspunkt der ICP-Anwendung und dient als Anmeldemaske.

Abbildung 2: LoginScreen

Funktionalität:

- Das Nickname-TextField kann einen beliebigen Nicknamen enthalten (fx:id `nicknameTextField`).
- Der Login-Button führt ein Login durch und wechselt auf den ChatScreen (fx:id `loginButton`).

Aufgabe 2.2 - ChatScreen

Im ClientScreen soll es möglich sein, mit anderen Benutzern zu chatten. Wird eine Nachricht mit selektiertem All-Tab gesendet, so erreicht diese Nachricht alle Benutzer, die online sind. Alternativ kann ein neuer Tab mit einem privaten Chat geöffnet werden, zum Beispiel durch das Doppelklicken eines Benutzers.

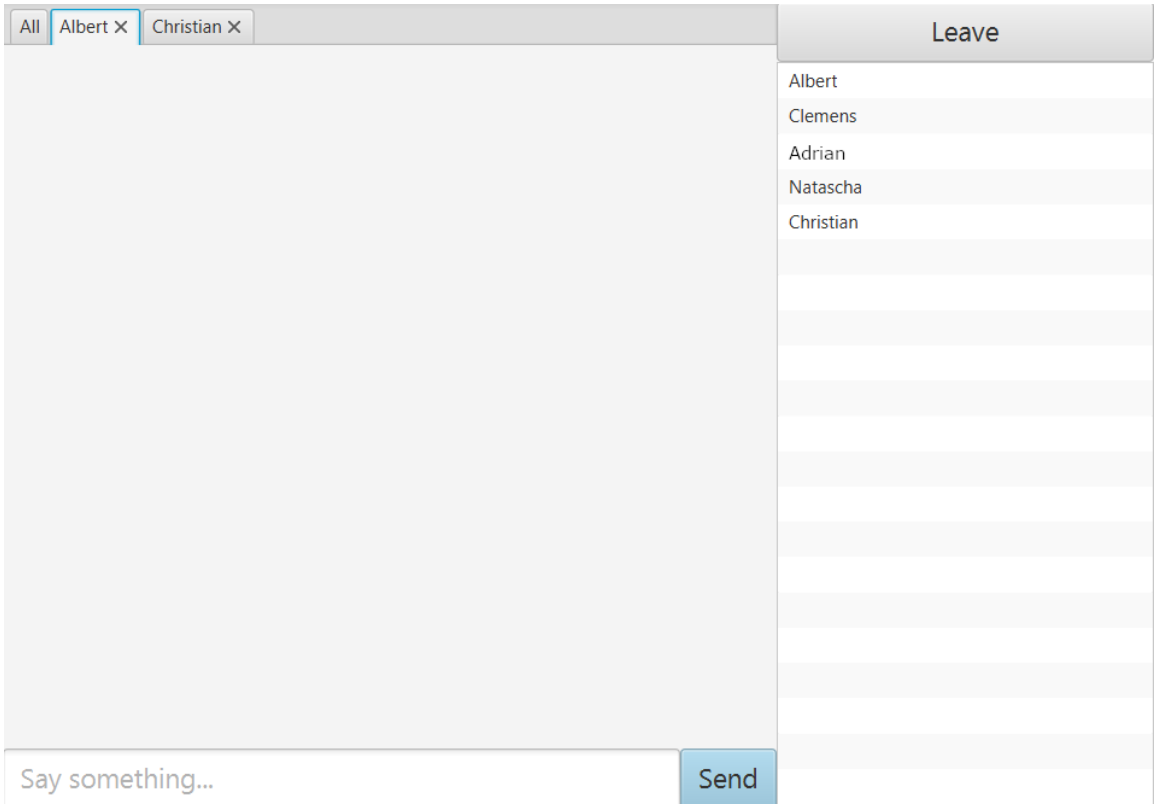

Abbildung 3: ChatScreen

Funktionalität:

- Zur Anzeige von verschiedenen Chats sollte eine TabPane (fx:id `chatTabPane`) verwendet werden. Hierbei gibt es einen All-Tab, welcher nicht geschlossen werden kann.
- Ein TextField dient zur Formulierung von Chatnachrichten (fx:id `messageTextField`).
- Ein Send-Button versendet eine Chatnachricht (fx:id `sendButton`). Optional kann dies zusätzlich über die Enter-Taste im Textfeld implementiert werden.
- Es existiert eine Liste aller momentan angemeldeten Benutzer, da nur mit diesen gechattet werden kann (fx:id `userListView`).
- Der Leave-Button führt einen Logout aus und öffnet den LoginScreen (fx:id `leaveButton`).

Aufgabe 3 - Funktionalität (8P)

Ziel dieser Aufgabe ist es, die Anwendung zu implementieren. Jede View wird dabei wie bekannt von einem eigenen Controller verwaltet. Zudem soll jeder Button mit Funktionalität versehen werden. In dieser Hausaufgabe soll noch keine Kommunikation über Websockets oder REST implementiert werden. Stattdessen werden Konsolenausgaben an den Stellen platziert, an denen später die eigentliche Implementierung erfolgt.

Beispiel:

Beim Klicken des Leave-Button wird auf der Konsole "Logout" ausgegeben.

Folgende Stichpunkte sollen als Hilfestellung für die zu implementierende Funktionalität dienen.

LoginScreen

- Login-Button -> Aktion auf Konsole ausgeben -> Scene wechseln

ChatScreen

- Leave-Button -> Aktion auf Konsole ausgeben -> Scene wechseln
- Send-Button -> Inhalt des TextFields auf Konsole ausgeben

Fenstertitel

Auch in dieser Anwendung sollen wieder Fenstertitel gesetzt werden. Erstelle die folgenden Konstanten mit den gegebenen Werten in deiner Konstanten-Klasse und verwende sie an den erforderlichen Stellen.

```
public static final String LOGIN_SCREEN_TITLE = "ICP Login";
public static final String CHAT_SCREEN_TITLE = "ICP Chat";
```

Aufgabe 4 - Tests (8P)

Wie in vorherigen Hausaufgaben soll die implementierte Funktionalität getestet werden. **Ein** GUI-Test soll **alle** in Aufgabe 2 beschriebenen Szenenwechsel prüfen. Konkret soll dieser Test alle Buttons klicken, Textfelder ausfüllen und Fenstertitel überprüfen. Es ist nicht notwendig die Konsolenausgaben zu testen.