Zürcher Hochschule
für Angewandte Wissenschaften

**School of Engineering**

InIT Institut für Angewandte Informations-technologie

# Wissenssicherung

## V1 – Verteilte Systeme

**Aufgabe V1.4:** Design UC03 – Leave a Chat

Erstellen Sie ein Design für die Realisierung der Systemoperation «leave(nickname: String, chatroomname: String = «Public»)», die das Hauptszenarios des Use Cases realisiert.

### Vorgehen

1. Entwerfen Sie die Systemoperation «leave(…)» und skizzieren Sie den Ablauf mit einem Kommunikationsdiagramm.
2. Aktualisieren Sie das DCD, so dass statisches und dynamisches Modell konsistent sind.

### Hinweise, Tipps

- Keine

### Ergebnis

Ein aktualisiertes DCD und ein Kommunikationsdiagramm für die entworfene Systemoperation.

**Zeit:** 15’

Zürcher Hochschule
für Angewandte Wissenschaften

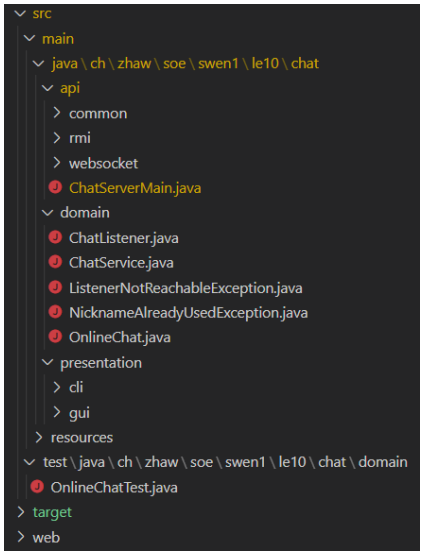
**School of Engineering**

InIT Institut für Angewandte Informations-technologie

**Aufgabe V1.5:** Implementierung des Online-Chats

Implementieren Sie die Domänenschicht für die Systemoperationen aus den Lernaufgaben (Aufgaben V1.2-V1.3) und der Wissenssicherung (Aufgabe V1.4) und testen Sie alles (s. Quellcode auf GitHub: https://github.zhaw.ch/SWEN1-LP2019/SWEN1_V1_Wissenssicherung_Verteilte_Systeme_V1.5-V1.6

Der abgegebene Quellcode hat folgende Package-Strukturierung:




Im Package «api» sind Implementierungen für die Kommunikation mit RMI und WebSocket. Im Package «presentation» sind ein JavaFX- und CLI-Client implementiert. Ein Web-Client ist im Folder «web» abgelegt. Alles in diesen Packages ist lauffähig und muss nicht angepasst werden. Ihre Aufgabe ist es, im Package «domain» den Online-Chat gemäss Ihrem Design zu implementieren. Dabei sollten Sie sich an das Interface «ChatService» halten, dass die öffentliche Schnittstelle für die Clients definiert.

## Vorgehen

1. Studierenden Sie den abgegebenen Quellcode im Package «domain». In diesem Package ist schon eine Skeleton-Klasse für den Online-Chat vorhanden (mit TO-DOs).
2. Implementieren Sie anhand Ihres Designs die drei Systemoperationen «join(…)», «leave(..)» und «post)».
3. Testen Sie die Systemoperationen mit ein paar angemessenen Testfällen (Unit-Tests).
4. Testen Sie die ganze Applikation mit dem mitgelieferten Server (Klasse «ChatServerMain» im Package «api») und den RMI-Clients (Packages «presentation.cli» und «presentation.gui»).

### Hinweise, Tipps

- Der Server kann einfach gestartet werden mit der Batch-Datei bzw. Skript `server`.
- Die Clients können mit den Batch-Dateien bzw. Skripts `client1` und `client2` gestartet werden.

### Ergebnis

Die implementierten und getesteten Systemoperationen und eine lauffähige Online-Chat-Applikation.

**Zeit:** 1 – 2 h

**Aufgabe V1.6:** WebSocket für die Kommunikation

Im Package «api.websocket» ist eine Implementierung des Online-Chats mit der Kommunikation über WebSockets und einer einfachen Webapplikation vorhanden.

Testen Sie Ihrer Lösung aus der Aufgabe V1.5 für eine zusätzliche Verwendung des Online-Chats über das Web.

Damit soll gezeigt werden, dass der Online-Chat einfach mit einem anderen Kommunikationsprotokoll und einem Web-Client erweitert werden kann.

### Vorgehen

1. Studieren Sie den Code im Package «api.websocket» und dem Verzeichnis «web» und insbesondere was für Design Patterns angewendet wurden.
2. Integrieren Sie die Anbindung Ihres Online-Chats (Domänenschicht).
3. Testen Sie die Anbindung mit der vorhandenen Webapplikation.

### Hinweise, Tipps

- Als Webserver wird ein eingebetteter Tomcat verwendet. Der Webserver kann einfach gestartet werden mit der Batch-Datei bzw. Skript `server`.
- Ein Web-Client (im Browser) wird aufgerufen mit der URL: `http://localhost:8080/ws/`.

### Ergebnis

Die getestete Webapplikation mit der implementierten Anbindung des Online-Chats.

**Zeit:** 30’

Die Besprechung dieser Aufgaben findet am Anfang des nächsten Unterrichtsblockes statt