Bewertung Übung «Chat»
🎖 Die Note ist: 5.63 (18.5 Punkte)

👥 👥 marcela.ruiz@zhaw.ch & Students: Schrom01-Fassband-Brandleo;

https://github.zhaw.ch/PM2-IT21bWIN-ruiz-mach-krea/Uebung-hk1-Schrom01-Fassband-Brandleo

Allgemeine Anforderungen (all-or-nothing)

> Voraussetzung für Punkterteilung: Die Applikation ist mit `gradle run` lauffähig.
- The server starts with Gradle run. Four clients were successfully started as well with gradle run. Very well done.

Software Grundfunktionalität

Funktionalität
🤖 SWGrund: 3: (0-3): 3

> * *Ihr Programm wird geladen und richtig "konfiguriert"*
> * *Der Server muss beliebig viele clients parallel bedienen können*
> * *Client Verbindung wird korrekt gemanagt (Erstellung, Cleanup)*
> * *Reihenfolge der Nachrichtenanmeldung wurde beibehalten*
> * *Meldung mit einem @Benutzername*

feedback:
- The program is properly configured. It loads in a proper way, it is well organised, and the readme file is very clear and helpful to run the application.
- The server is able to serve serveral clients. In a test with 4 clients it worked very well.
- The client connection is well handled. The connection and disconnection of clients is registered in the console.
- The order of message login is maintained.
- It is possible to send messages to specific clients by means of "@". The server detects when a message is sent to a client that has not been registered. This is recorded in the console and shown in the UI. In addition, a client can send a message to itself. This is properly handled.
- Very good, the application automatically delete messages that have been already sent. That help users to better interact with the UI. Well done.

MVC
🤖 SWMVC: 1: (0-1): 0.5

> MVC/MVP wurde korrekt umgesetzt, oder im Dokumentation eine Gegenposition dargelegt

feedback:
- Currently the model and controller are part of the same class "chat window controller". For instance, for the method connect(), a way to implement the MVC could look like:

```
private void connect() {
        try {
            model.clearMessages();
            startConnectionHandler();
            connectionHandler.connect();
        } catch(ChatProtocolException | IOException e) {
            writeError(e.getMessage());
        }
    }
```

This would mean that we have a chatClientModel that implements clearMessages(). For instance:

```
 public void clearMessages() {
        messageList.clear();
    }
```

- Despite the application works, it does not have the MVC pattern implemented.
- On a positive note, there is not duplicated code. This aspect has been observed.
- The protocol clases are OK.

GUI

SWGui: 1: (0-1): 1

> - GUI verhindert unzulässige Operationen
> - Sinnvolle Statusmeldungen über Konsole
> - Der Filter funktioniert im GUI und Deaktivieren des Filters zeigt alle Nachrichten

feedback:
- The GUI works properly, and meaningful messages are sent via the console.
- The Filter works properly. When it is deactivated all messages are shown.
- In general the GUI works as expected. As mentioned before, it would be positive to clean up sent messages.

Cleancode

CleanCode L1

CcL1: 3: (0-3): 3

> - Namenskonventionen eingehalten
> - Namen sind sinnvoll gewählt
> - Code ist aufgeräumt (Reihenfolge, Leerzeilen, Einrücken, kein toter Code)
> - Code ist übersichtlich gehalten (Struktur)
> - _*Methodengestaltung*_ Länge
> - Klassen haben Verhalten (L1)
> - Ihre Klassen stehen für sich alleine, siehe Klassendiagramm.
> - JavaDoc vorhanden
> - Zugriffsmodifikatoren (public, private, protected, keiner) sind sinnvoll gewählt

feedback:
- In general the clean code level L1 has been observed. Names are chosen sensibly, code is tidy, Gradle has been used for structuring the code, methods length and structure is well balanced. JavaDoc is present. Access modifiers are OK. Class diagram shows a proper cohesion among classes.

CleanCode L2

CcL2: 1: (0-1): 1

> - Argumente werden überprüft, Exceptions werden behandelt
> - Ausnahmen werden behandelt (L2)
> - Klassen stehen für sich alleine

feedback:
- Arguments are checked and there is exceptions handling.

CleanCode L3

CcL3: 1: (0-1): 1

> - Lange Parameterlisten durch Builder oder Argumentobjekt ersetzen (L3)
> - Vernünftige Objektlebenszeiten (L3) connections werden geschlossen
> - Anwender erhält sinnvolle Fehlermeldung (L3)

feedback:
- Methods are passing objects. Sensible error messages are shown to users. The clean code level 3 is observed.

Dokumentation

Dokumentation, Beschreibung der Probleme

Dok1: 4: (0-4): 3

> - Die gefundenen Probleme wurden in GitHub Issues dokumentiert
> - Es wurde unterschieden zwischen funktionalen und strukturellen Problemen
> - Es wurden mindestens je 5 funktionale und 5 strukturelle Probleme beschrieben
> - Die Problembeschreibung ist verständlich und nachvollziehbar
> - Strukturelle Probleme zeigen Lösungsoptionen und Begründen die gewählte Lösungsoption

> - Funktionale Fehler sind mit Anleitung zur Reproduktion beschrieben

feedback:
- The students have several issues on GitHub.
- A distinction was made betwen functional and structural problems. This distinction is sufficient. In total 10 problems have been identified. All of them have been described and linked in the Readme. Several issues have been created for each problem. In between 1 to 3 issues per problem.
- In various cases, the problem description could be more detailed. In addition, the students do not detail the reasoning behind the selected solution. For instance, what kind of reasoning was implemented behing the MVC? What kind of MVC was applied for the server? What was the restructuring performed for the protocol? Descriptions are too abstract.

Dokumentation Github

Dok2: 1: (0-1): 1

> - Die Commits sind korrekt mit den Issues verknüpft
> - Die Issues sind im Readme verlinkt und gruppiert dargestellt

feedback:
- Regarding GitHub, the commits are properly linked to the issues. In addition, issues are linked and grouped under problems in the readme file.

Dokumentation, Beschreibung der Lösung

Dok3: 4: (0-4): 4

> - Es ist eine Erläuterung der Lösung vorhanden (mindestens 0.5 Seiten)
> - Die Beschreibung der Lösung ist fachlich und inhaltlich korrekt
> - Die Erklärung zur Lösung ist nachvollziehbar
> - Die Aufteilung der Klassen in Packages ist sinnvoll.
> - Gemeinsame Funktionalität {Server,Client}ConnectionHandler in Superklasse ausgelagert
> - Die Klassenstruktur ist sinnvoll, die Koppelung ist niedrig, z. B. ClientMessage und Paket wurden erstellt.

feedback:
- An explanation of the solution is presented in the readme. The solution is correct and the information is comprehensible.
- The classes are properly divided into packages.
- The class structure is correct. Packages are used.
- Code documentation could improve.

Dokumentation Klassendiagramm

Dok4: 1: (0-1): 1

> - Klassendiagramm vorhanden
> - Klassendiagramm übersichtlich, genügend detailliert und hält sich an die Notation

feedback:
- The class diagram is appropriate, it has a proper notation, and summarises the application in a sensible manner.