Protokoll Chat für Schwerhörige

René Pöcher, René Hollander

November 20, 2014

Contents

1 Aufgabenstellung

S04: Chat für Schwerhörige

Aufgabe für 2 Personen

Erstellt ein einfaches Chat-Programm für "Schwerhörige", mit dem Texte zwischen zwei Computern geschickt werden können.

Dabei soll jeder gesendete Text "geschrien" ankommen (d.h. ausschließlich in Großbuchstaben, lächelnd wird zu *lol*, Buchstaben werden verdoppelt, ... - ihr dürft da kreativ sein)

Zusätzlich sollen "böse" Wörter ausgefiltert und durch "$%&*" ersetzt werden. Diese Funktionalität soll aber im Interface jederzeit aktiviert und deaktiviert werden können.

Verwende dafür ausgiebig das Decorator-Pattern.

2 Designüberlegung

1) Verwenden Sie das Decorator-Pattern:

Das Message Objekt wird als Core verwende. Die verschiedenen Zusatzfunktionen werden in den Decorator Klassen implementiert. So können Zusatzfunktionen jederzeit ausgetauscht, abgeschaltet oder hinzugefügt werden

2) Der Chat

Der Nickname wird mit der Nachricht mitgeschickt, so können die verschiedenen Clients sich von einander unterscheiden. Als Kommunikationsmittel wird Multicast Sockets (UDP) verwendet.

Die Nachricht wird binär übertragen:
4 Byte Länge des Nickname
Nickname UTF-8 kodiert
4 Byte Länge der Nachricht
Nachricht UTF-8 kodiert

Als Puffer werden auf der Empfängerseite 4KB verwendet.

```
private class SocketReader implements Runnable, Closeable {

    final Logger LOGGER = LogManager.getLogger(SocketReader.class);

    private MulticastSocket socket;
    private Handler<Message> handler;

    private boolean running;
    private Thread thread;

    public SocketReader(MulticastSocket socket, Handler<Message> handler) {
        this.socket = socket;
        this.handler = handler;

        this.running = true;
        this.thread = new Thread(this);
        this.thread.start();
    }
```

```
public ChatClient(InetAddress groupAdress, int port, Handler<Message> handler) throws IOException {
    this.groupAdress = groupAdress;
    this.port = port;
    this.setHandler(handler);

    this.socket = new MulticastSocket(this.port);
    this.socket.setReuseAddress(true);
    this.socket.joinGroup(this.groupAdress);
    this.socketReader = new SocketReader(this.socket, this.handler);
}
```

3 Zeitaufstellung

3.1 Zeitschätzung

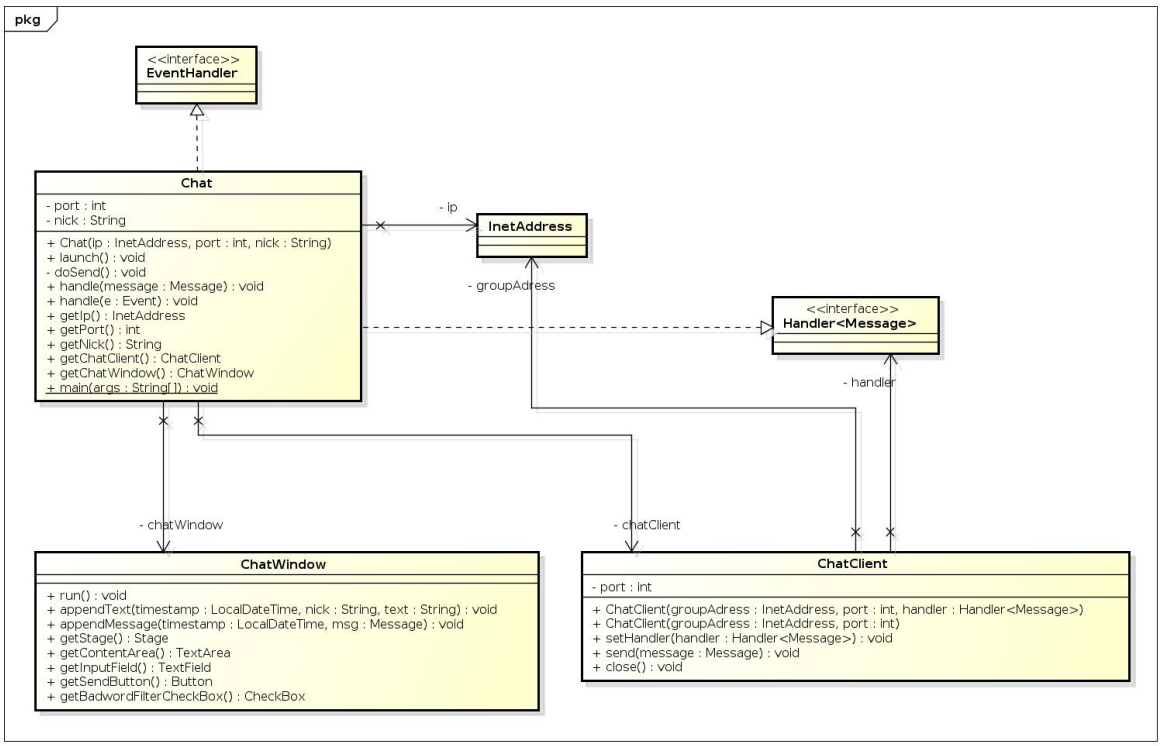
180 Minuten Implementieren und Kommentieren pro Person
45 Minuten Testen pro Person
30 Minuten Protokoll pro Person

3.2 Zeitaufzeichnung

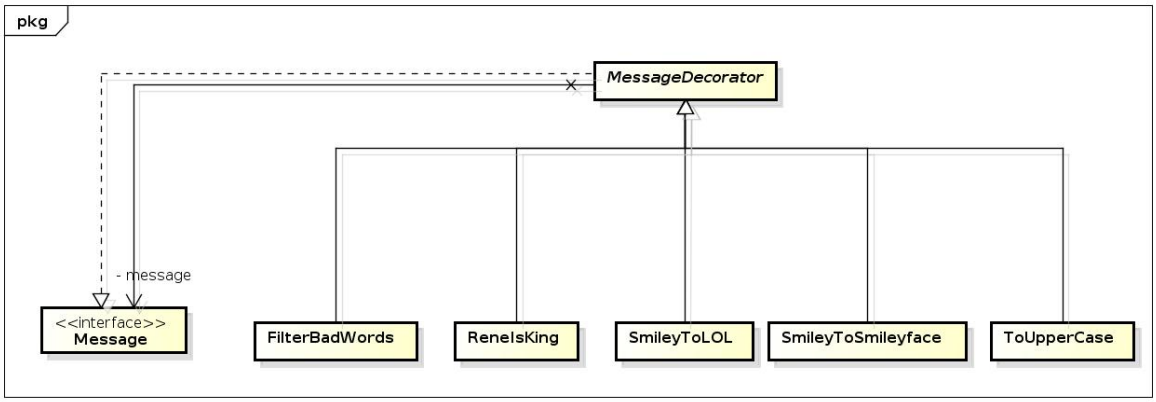
Rene Hollander	Implementation GUI 2	30 Minuten
Rene Hollander	Implementation ChatClient	60 Minuten
Rene Hollander	Testen und Javadoc vervollständigen	45 Minuten
Rene Hollander	Erste Version des Protokoll	20 Minuten
Rene Pöcher	Dekorator designen	30 Minuten
Rene Pöcher	Implementation Message/MessageDecorator	65 Minuten
Rene Pöcher	Zusätzliche Dekorator Funktionen implementieren	30 Minuten
Rene Pöcher	Dokumentieren und finales Protokoll	15 Minuten

4 UML

4.1 UML



4.2 UML-Decorators



5 Arbeitsdurchführung/Lessons Learned

Für eine einfache Umsetzung des Chatraumes wurde Multicast verwendet. Ein Multicast Socket kann einer Gruppe beitreten (Multicast Addresse) und empfängt dann alle gesendeten Nachrichten.Möchte man in einen anderen Chatraum, nimmt man einfach eine andere IP. Ein Nachteil von Multicast ist, das es meist nur im eigenen Netzwerk funktioniert.

Für eine schnelle Programmierung ist es am besten die verschiedenen Funktionen herauszusuchen und dementsprechend zu verlinken. So müssen nicht alle Programmteile selber programmiert/designet werden.

6 Quellen

[1] ReplaceAll Funktion
Link: http://stackoverflow.com/a/16574312
Zuletzt abgerufen am: 20.11.2014

7 Git:

https://github.com/ReneHollander/au04-chat.git