Chat für Schwerhörige

Kodras, Ritter

2014

4AHIT

13.11.2014

Inhalt

[Aufgabenstellung 4](#_Toc404287045)

[Designüberlegung 5](#_Toc404287046)

[Aufwandschätzung 6](#_Toc404287047)

[Testbericht 6](#_Toc404287048)

[GitHub-Link 6](#_Toc404287049)

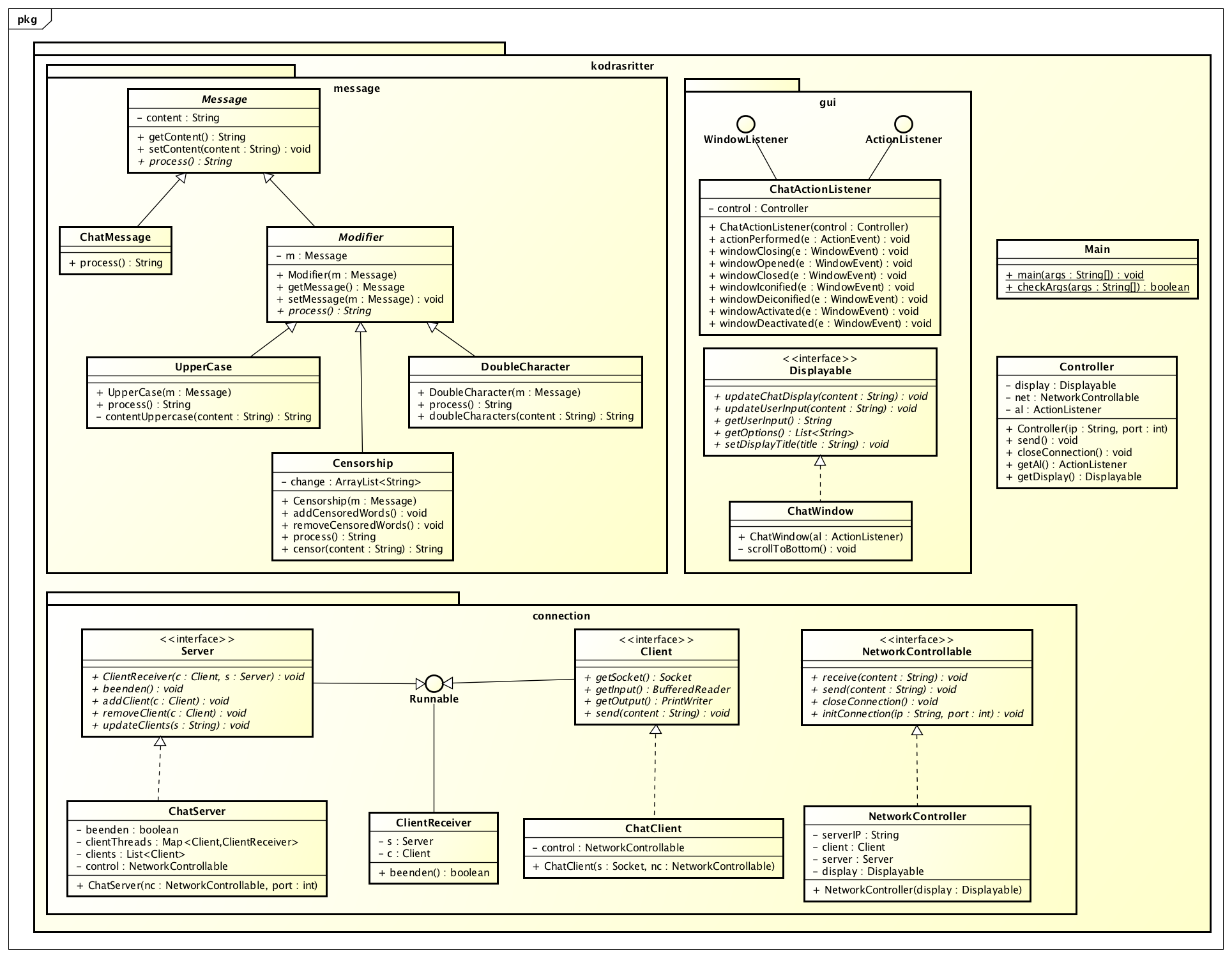
# Aufgabenstellung

**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.  
  
Nähere Informationen zum Transport von Daten über das Netzwerk findet ihr [hier](http://docs.oracle.com/javase/tutorial/networking/overview/networking.html).

Fock, KritzlEl-Far, Weinberger  
Ableitinger, Erceg  
Kodras, Ritter  
Hamberger, Hampl  
Schober, Wortha  
Jevtic, Malik  
Kölbl, Steinkellner  
Kalauner, Tiryaki  
Geyer, Polydor  
Stokic, Taschner  
Hollander, Pöcher  
Hackenberger, Kocsics

# Designüberlegung



Das Klassendiagramm wurde zur besseren Ansicht zusätzlich als PNG-File im gleichen Ordner wie dieses Dokument gespeichert.

Das Decorator-Pattern wurde verwendet, um eine Chat-Nachricht mit einem bestimmten Modifier (UpperCase, Censorship und DoubleCharacter) zu dekorieren. Die Funktionen können einzeln aktiviert und wieder deaktiviert werden. Beim „Einpacken“ der einzelnen Komponenten werden die ausgewählten Optionen berücksichtigt.Aufwand

## Schätzung & Aufteilung

**Geschätzter Aufwand:** 5h \* 2 Personen = 10 Stunden

|  |  |
| --- | --- |
| **Aufgabe** | **Name** |
| UML-Klassendiagramm | Kodras, Ritter |
| Implementierung Package kodrasritter | Ritter |
| Implementierung Package kodrasritter.connection | Ritter |
| Implementierung Package kodrasritter.gui | Kodras |
| Implementierung Package kodrasritter.message | Kodras |
| Testen | Kodras, Ritter |
| Protokoll fertigstellen | Kodras, Ritter |

## Reeller Aufwand

|  |  |  |
| --- | --- | --- |
| **Aufgabe** | **Name** | **Zeit (min)** |
| UML-Klassendiagramm | Kodras | 100 |
| UML-Klassendiagramm | Ritter | 100 |
| Implementierung Package kodrasritter | Ritter | 50 |
| Implementierung Package kodrasritter.connection | Ritter | 180 |
| Implementierung Package kodrasritter.gui | Kodras | 45 |
| Implementierung Package kodrasritter.gui | Ritter | 60 |
| Implementierung Package kodrasritter.message | Kodras | 60 |
| Testen | Kodras | 30 |
| Testen | Ritter | 80 |
| Protokoll fertigstellen | Kodras | 30 |
| Protokoll fertigstellen | Ritter | 30 |

**Aufwand Kodras:** 265 Minuten: 4 Stunden 25 Minuten

**Aufwand Ritter:** 500 Minuten: 8 Stunden 20 Minuten

**Gesamtaufwand:** 765 Minuten: 12 Stunden 45 Minuten

# Implementierung/Durchführung

Folgende Klassen/Packages/Funktionalitäten wurden implementiert:

## Package kodrasritter

### Main

* Initialisiert einen neuen Controller
* Beinhält main-Methode
* Überprüft die Kommandozeilenargumente

### Controller

* Initialisiert ein neues Display zur Darstellung, Networkcontroller (Verbindungsaufbau) und Actionlistener
* Sendet Eingaben aus der GUI an den Networkcontroller, indem ein neues Message-Objekt erstellt wird, welches je nach gewählten Optionen dekoriert wird.

## Package kodrasritter.connection

### Networkcontrollable

* Initialisiert einen neun Client und falls nicht vorhanden einen neuen Server (Verbindungen über Sockets)
* Sendet Nachrichten an den Server
* Empfänger Nachrichten vom Server und stellt sie am Display dar
* Schließt die Verbindung

### Networkcontroller

* Implementiert das Interface Networkcontrollable

### Client

* Sendet Nachrichten an den Server
* Empfängt Nachrichten vom Server (nebenläufig)

### ChatClient

* Implementierung von Client

### Server

* Akzeptiert neue Clients
* Empfängt Nachrichten der Clients, die sich beim Server anmelden (siehe ClientReceiver)
* Sendet empfangene Nachrichten an alle registrierte Clients

### ChatServer

* Implementierung von Server

### ClientReceiver

* Empfängt Nachrichten eines Clients (nebenläufig)

## Package kodrasritter.message

### Message

* Stellt eine Nachricht dar und enthält deren Inhalt
* Enthält eine Abstrakte Methode process zur Verarbeitung des Nachrichteninhalts

### ChatMessage

* Erbt von Message
* Implementiert die Verarbeitung

### Modifier

* Erbt von Message
* Enthält eine weitere Message

### UpperCase

* Erbt von Modifier
* Verarbeitet den Inhalt der Nachricht, indem die Buchstaben groß geschrieben werden

### DoubleCharacter

* Erbt von Modifier
* Verarbeitet den Inhalt der Nachricht, indem die Buchstaben verdoppelt werden

### Censorship

* Erbt von Modifier
* Verarbeitet den Inhalt der Nachricht, indem „Bad Words“ zensiert werden

### Displayable

* Updatet das Display & den User-Input
* Gibt Auskunft über vom User ausgewählte Optionen sowie den Userinput

### ChatWindow

* Implementierung von Displayable

### ChatActionListener

* Implementierung eines Action- & Window-Listeners

# Testbericht

Die Tests wurden mit JUnit durchgeführt. Eine genaue Beschreibung der Testfälle findet man in den Javadocs. Alle Testfälle (40) wurden erfolgreich durchlaufen.

Test Coverage: ca. 95%

# GitHub-Link

<https://github.com/dkodras01-tgm/SEW-S04.git>