S04 – Chat für Schwerhörige

Stokic Stefan, Taschner Thomas

2014

Inhaltsverzeichnis

[Aufgabenstellung: 2](#_Toc404424849)

[Designüberlegung: 2](#_Toc404424850)

[Arbeitsschritte: 2](#_Toc404424851)

[Zeitaufwand: 3](#_Toc404424852)

[Arbeitsdurchführung: 4](#_Toc404424853)

[Testdurchläufe: 5](#_Toc404424854)

[Quellenangaben: 6](#_Toc404424855)

## Aufgabenstellung:

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: <https://docs.oracle.com/javase/tutorial/networking/overview/networking.html>

## Designüberlegung:

<< Hier bitte das UML >>

## Arbeitsschritte:

* ChatConnection
* ChatListener
* MulticastChat
* Message
* ChatMessage
* TypeMessage
* CapsedMessage
* BadwordFilteredMessage
* EnterListener
* GUIWindowListener
* MultiCastChatMenu
* MultiCastChatClient
* Main (nur Starter Klasse)
* ChatTest (Testing)

## Zeitaufwand:

|  |  |  |
| --- | --- | --- |
| **Thema** | **Autor** | **Zeitaufwand** |
| ChatConnection | Stokic | 2min |
| ChatListener | Stokic | 2h |
| MulticastChat | Stokic, Taschner | 6h |
| Message | Stokic | 10min |
| ChatMessage | Stokic | 2min |
| TypeMessage | Stokic | 10min |
| CapsedMessage | Stokic | 15min |
| BadwordFilteredMessage | Taschner, Stokic |  |
| EnterListener | Taschner |  |
| GUIWindowListener | Taschner, Stokic |  |
| MultiCastChatMenu | Taschner, Stokic |  |
| MultiCastChatClient | Taschner, Stokic |  |
| Main | Taschner |  |
| ChatTest | Taschner |  |
| UML |  |  |

## Arbeitsdurchführung:

* **Erfolge:**
  + Das richtige Anwenden des Decorator -und Singletons-Pattern.
  + Implementierung des ChatListeners und des MulticastChats.
  + GUI
* **Schwierigkeiten:**
  + Es gab Schwierigkeiten bzw. Denkfehler, als wir die Empfangenen Nachrichten erhalten haben, diese in der Textarea auszugeben. Am Anfang war es etwas schwer da es nicht wirklich entkoppelt war, aber durch die Hilfe des Singleton-Patterns konnten wir unser Problem lösen.
  + Das closen eines MulticastSockets führte immer zu einem Fehler weshalb wir deshalb socket.close() auskommentiert haben und nicht verwenden.
  + Wir haben uns lange Gedanken darüber gemacht wie wir das Decorator-Pattern und wo verwenden. Am Ende haben wir uns entschieden das wir ein Message-Objekt machen(abstrakt) und davon aus, normale Messages, Capslocked und Badwordfiltered Messages, haben. Das Decorator-Pattern konnte hier sehr gut angewandt werden.

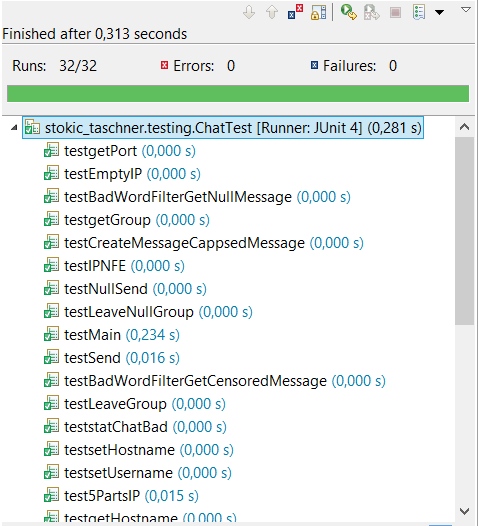
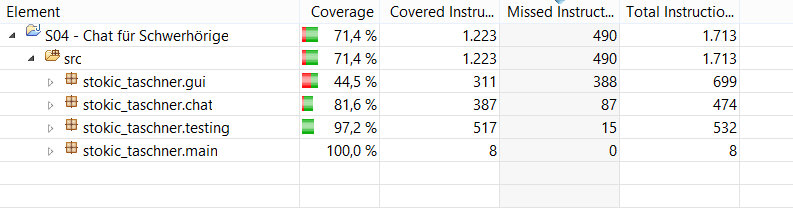
## Testdurchläufe:

Tests konnten wir mithilfe von JUnit durchführen.

Die Test-Coverage wurde insgesamt zu 71,4% abgedeckt.

Da wir noch keine GUI-Tests im Unterricht vorgenommen haben, wurden GUI-Tests ausgelassen was somit führte das im gui-package nur 44,5% der Test-Coverage abgedeckt. Das chat-package konnte zu 81,6% abgedeckt werden und das testing-package selbst, zu 97,2%.

Alle 32 Test Durchläufe sind erfolgreich.



## Quellenangaben:

<https://docs.oracle.com/javase/tutorial/networking/overview/networking.html>

<https://docs.oracle.com/javase/tutorial/networking/datagrams/broadcasting.html>

<http://java.kompf.de/multicast.html>

<https://docs.oracle.com/javase/7/docs/api/index.html?java/net/MulticastSocket.html>

<http://www.programcreek.com/2012/05/java-design-pattern-decorator-decorate-your-girlfriend/>

<http://www.theserverside.de/singleton-pattern-in-java/>