S04 – Chat für Schwerhörige

Stokic Stefan, Taschner Thomas

2014

Inhaltsverzeichnis

[Aufgabenstellung: 2](#_Toc404430759)

[Designüberlegung: 2](#_Toc404430760)

[Arbeitsschritte: 5](#_Toc404430761)

[Zeitaufwand: 6](#_Toc404430762)

[Arbeitsdurchführung: 7](#_Toc404430763)

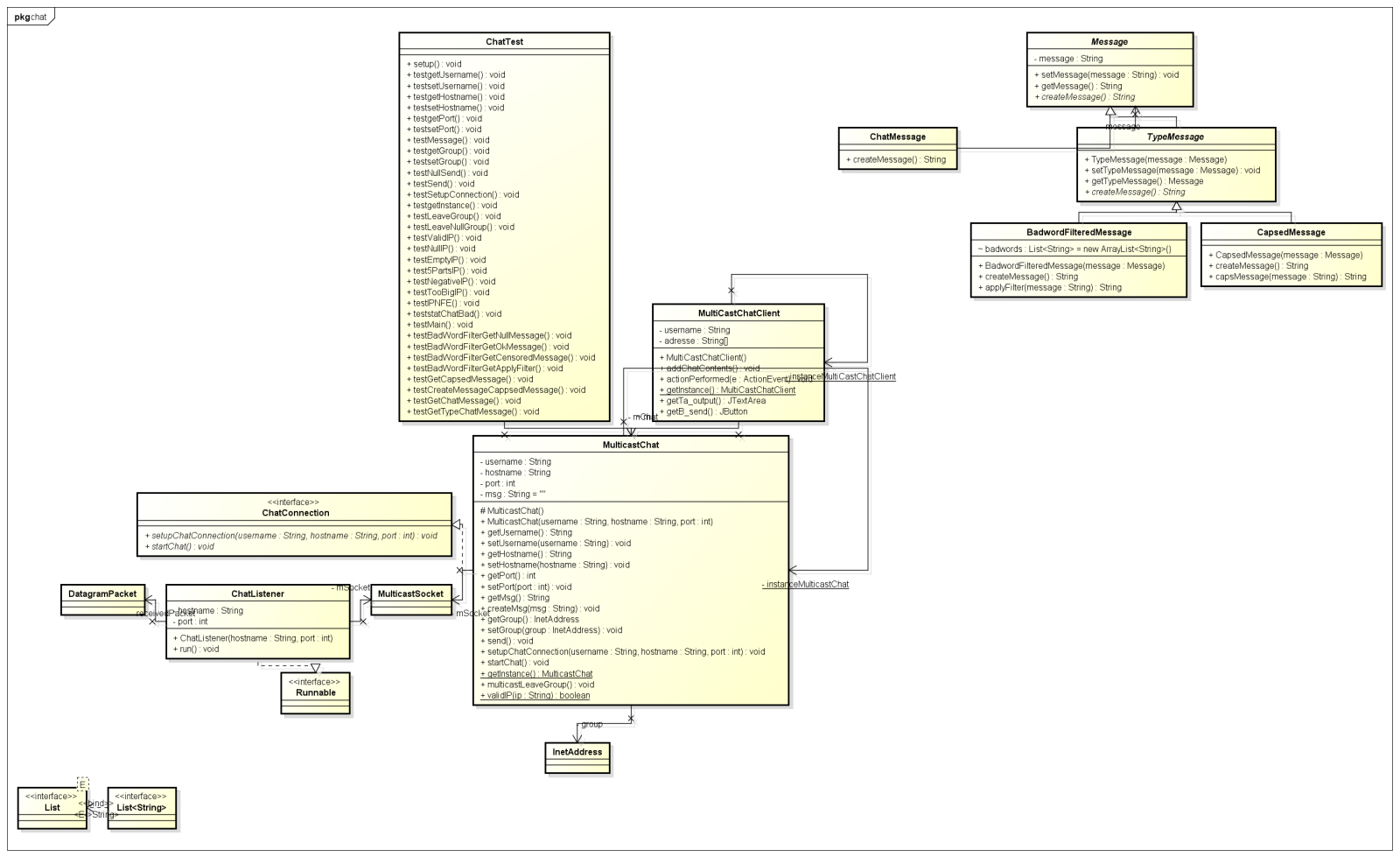
[Testdurchläufe: 8](#_Toc404430764)

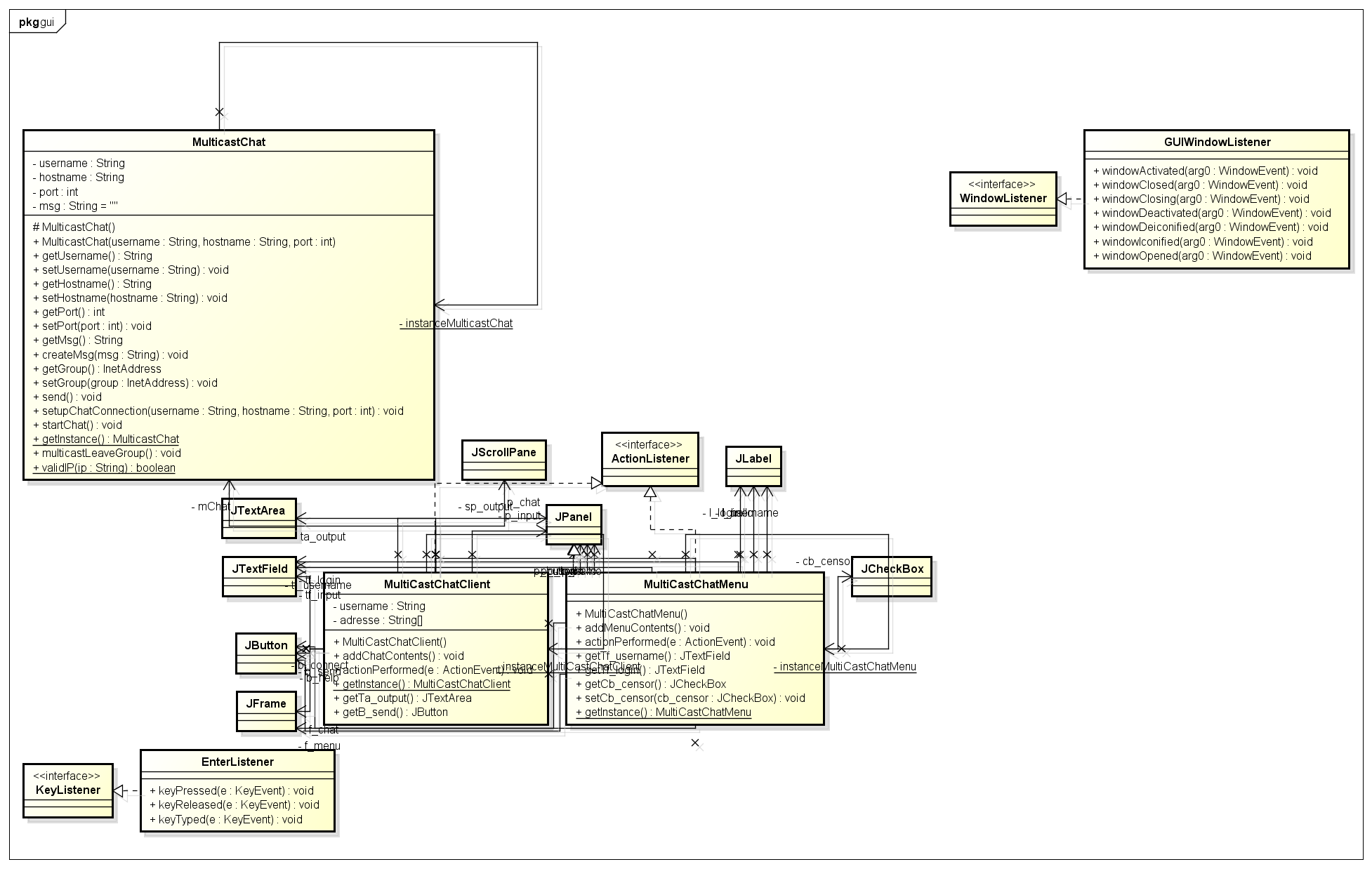
[Quellenangaben: 9](#_Toc404430765)

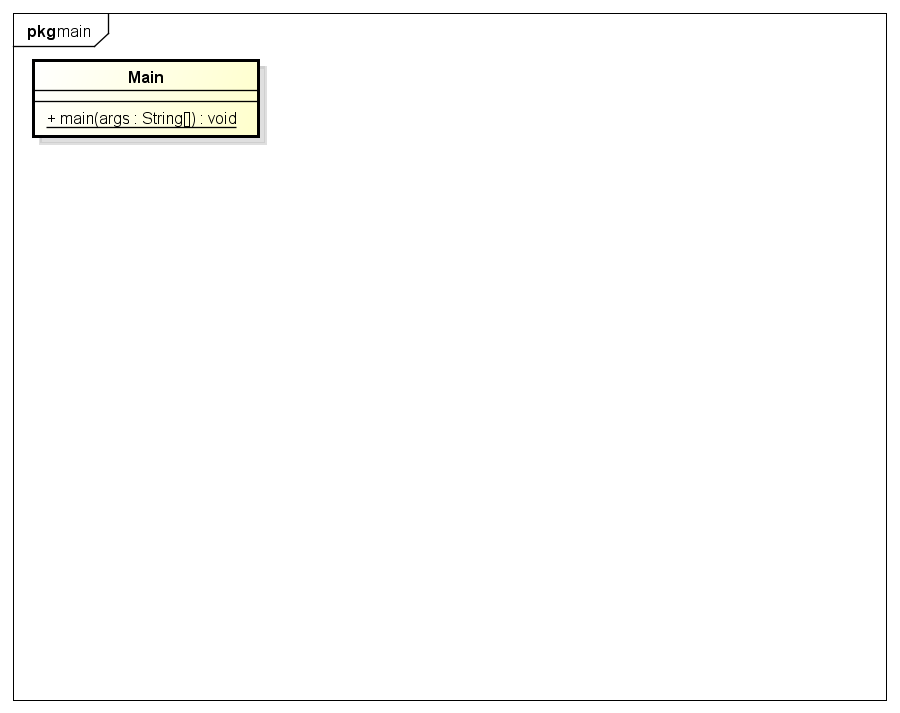
## Aufgabenstellung:

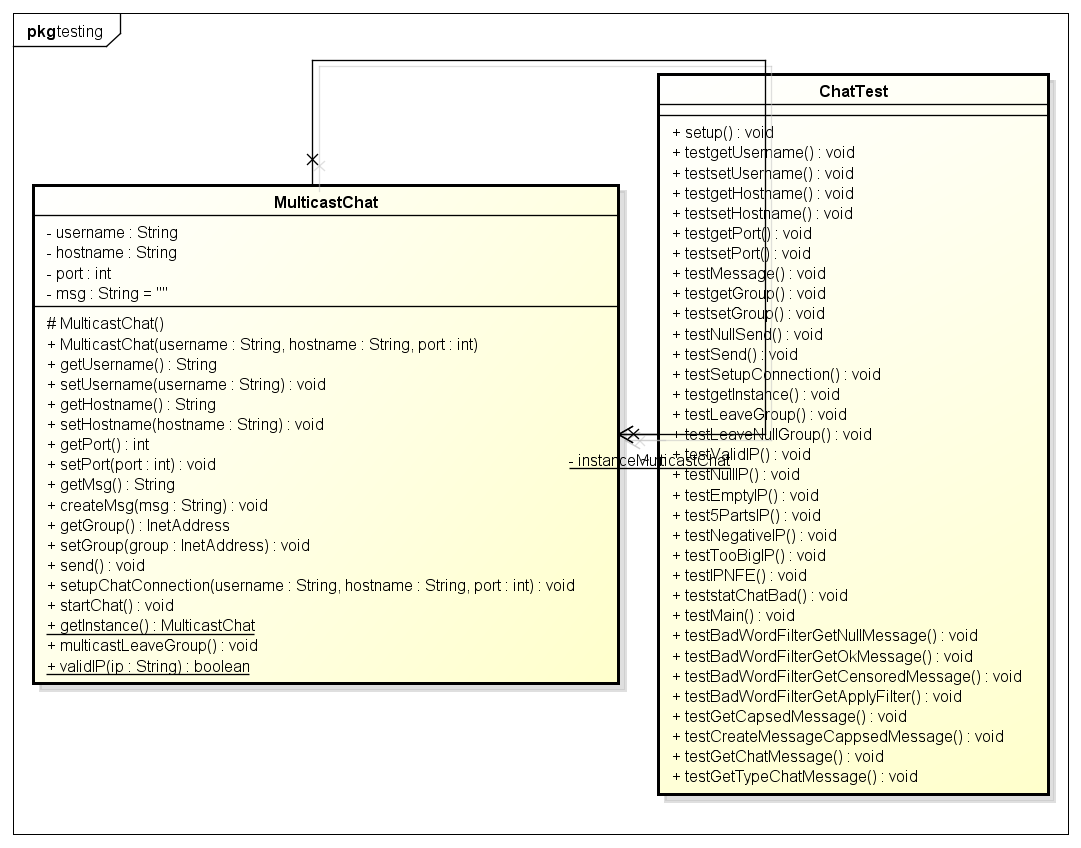
Aufgabe für 2 Personen  
  
Erstellt ein einfaches Chat-Programm für "Schwerhörige", mit dem Texte zwischen zwei Computern geschickt werden können.  
  
Dabei soll jeder gesendete Text "geschrien" ankommen (d.h. ausschließlich in Großbuchstaben, lächelnd wird zu \*lol\*, Buchstaben werden verdoppelt, … - 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:









## 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 | 20min |
| EnterListener | Taschner | 5min |
| GUIWindowListener | Taschner, Stokic | 5min |
| MultiCastChatMenu | Taschner, Stokic | 5h |
| MultiCastChatClient | Taschner, Stokic | 5h |
| Main | Taschner | 1min |
| ChatTest | Taschner | 5h |
| UML | Stokic, Taschner | 5min |

## Arbeitsdurchführung:

* **Erfolge:**
  + Das richtige Anwenden des Decorator- und Singleton-Pattern.
  + Implementierung des ChatListeners und des MulticastChats.
  + GUI
* **Schwierigkeiten:**
  + Es gab Schwierigkeiten bzw. Denkfehler, die empfangenen Nachrichten in der Textarea auszugeben. Am Anfang gab es aufgrund der fehlenden Kopplung noch Schwierigkeiten, durch die Hilfe des Singleton-Patterns konnten wir jedoch unser Problem lösen.
  + Das Schließen eines MulticastSockets führte immer zu einem Fehler, weshalb wir socket.close() auskommentiert und nicht verwendet haben.
  + Wir haben uns lange Gedanken darüber gemacht, wie wir das Decorator-Pattern wo anwenden könnten. Schlussendlich haben wir uns entschieden, dass wir ein abstraktes Message-Objekt erstellen und in Folge dessen normale Messages, Capslocked und Badwordfiltered Messages haben. Das Decorator-Pattern konnte hier sehr gut angewandt werden.

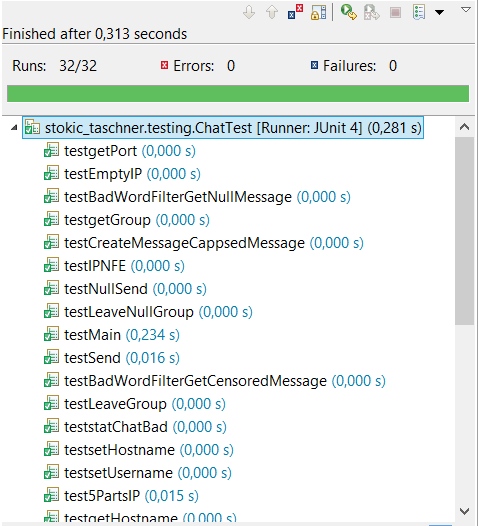
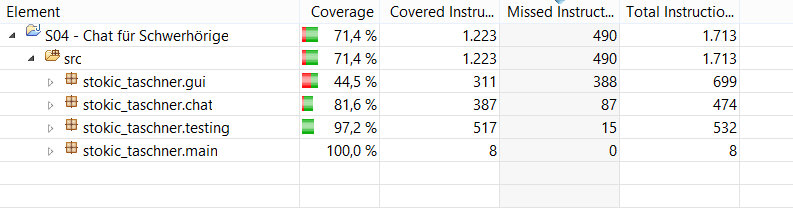
## Testdurchläufe:

Tests konnten wir mithilfe von JUnit durchführen.

Die Test-Coverage beträgt insgesamt 71,4%.

Da wir noch keine GUI-Tests im Unterricht vorgenommen haben, wurden GUI-Tests ausgelassen was somit zu einer Coverage von 44,5% im gui-package führte. 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/>