

S04 – Chat für Schwerhörige

Stokic Stefan, Taschner Thomas

Inhaltsverzeichnis

Aufgabenstellung:	2
Designüberlegung:	2
Arbeitsschritte:	5
Zeitaufwand:	6
Arbeitsdurchführung:	7
Testdurchläufe:	8
Quellenangaben:	9

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)

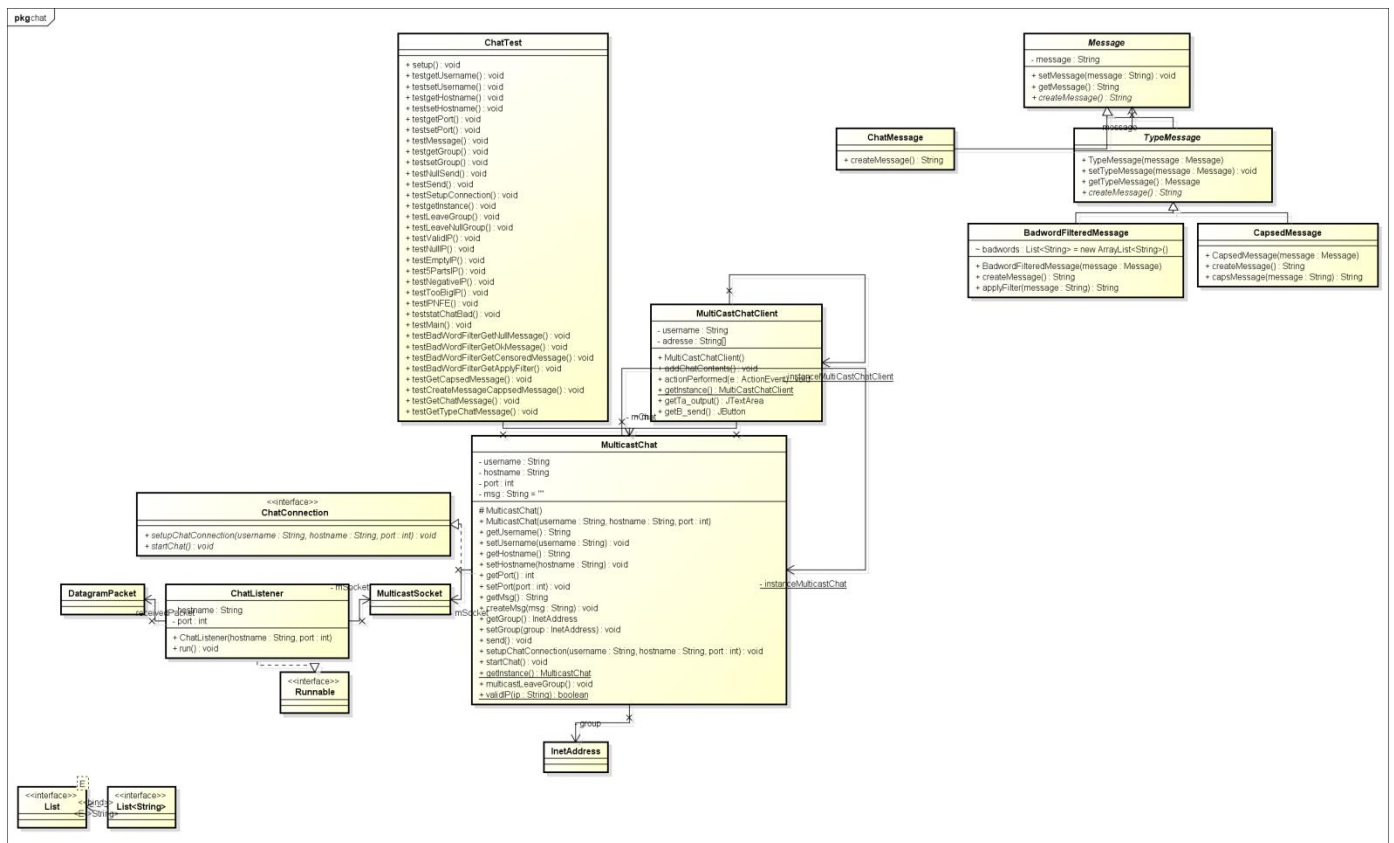
Zusätzlich sollen "böse" Wörter ausgefiltert und durch "\$%&*" ersetzt werden. Diese Funktionalität soll aber im Interface jederzeit aktiviert und deaktiviert werden können.

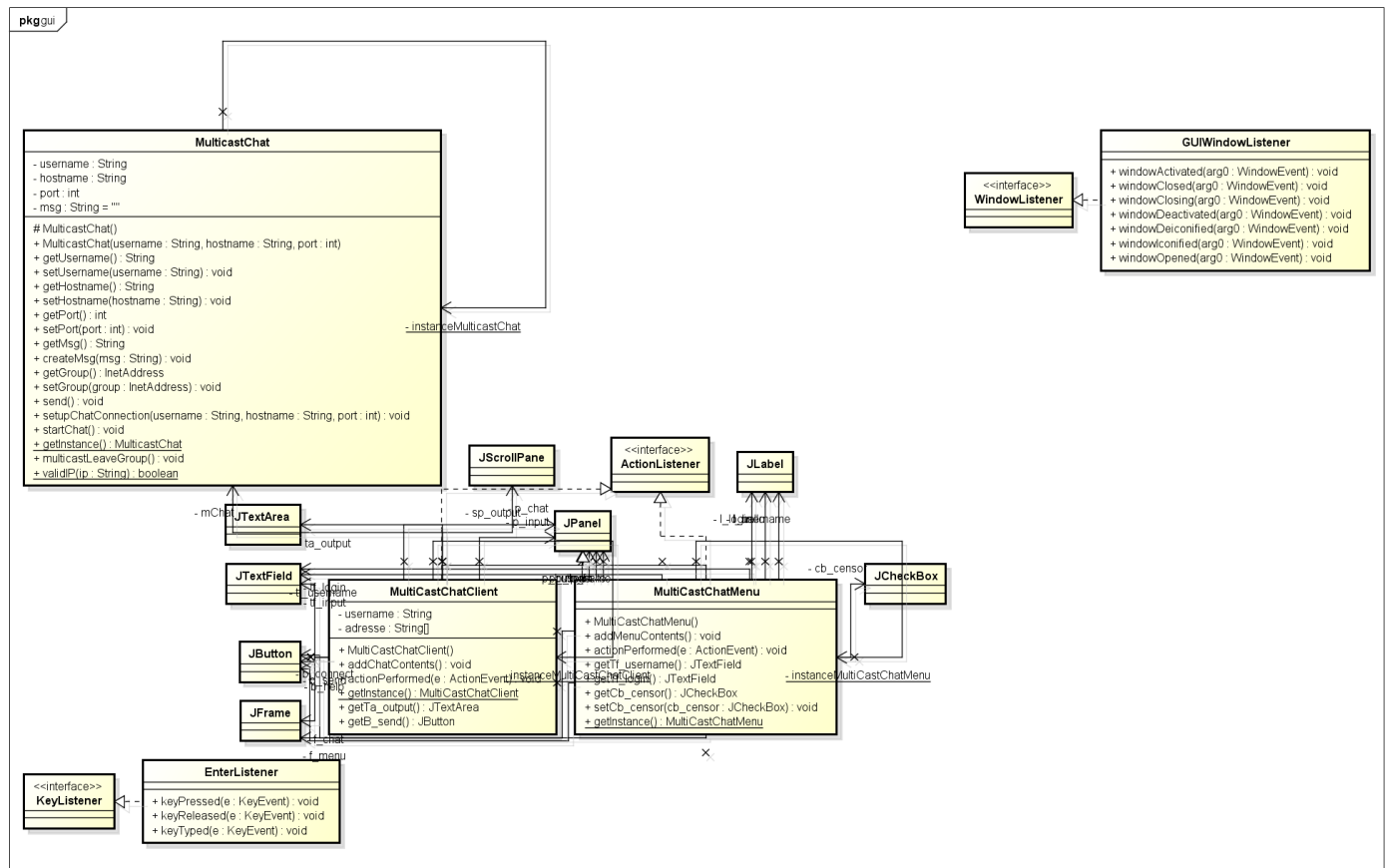
Verwende dafür ausgiebig das Decorator-Pattern.

Nähere Informationen zum Transport von Daten über das Netzwerk findet ihr hier:

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

Designüberlegung:

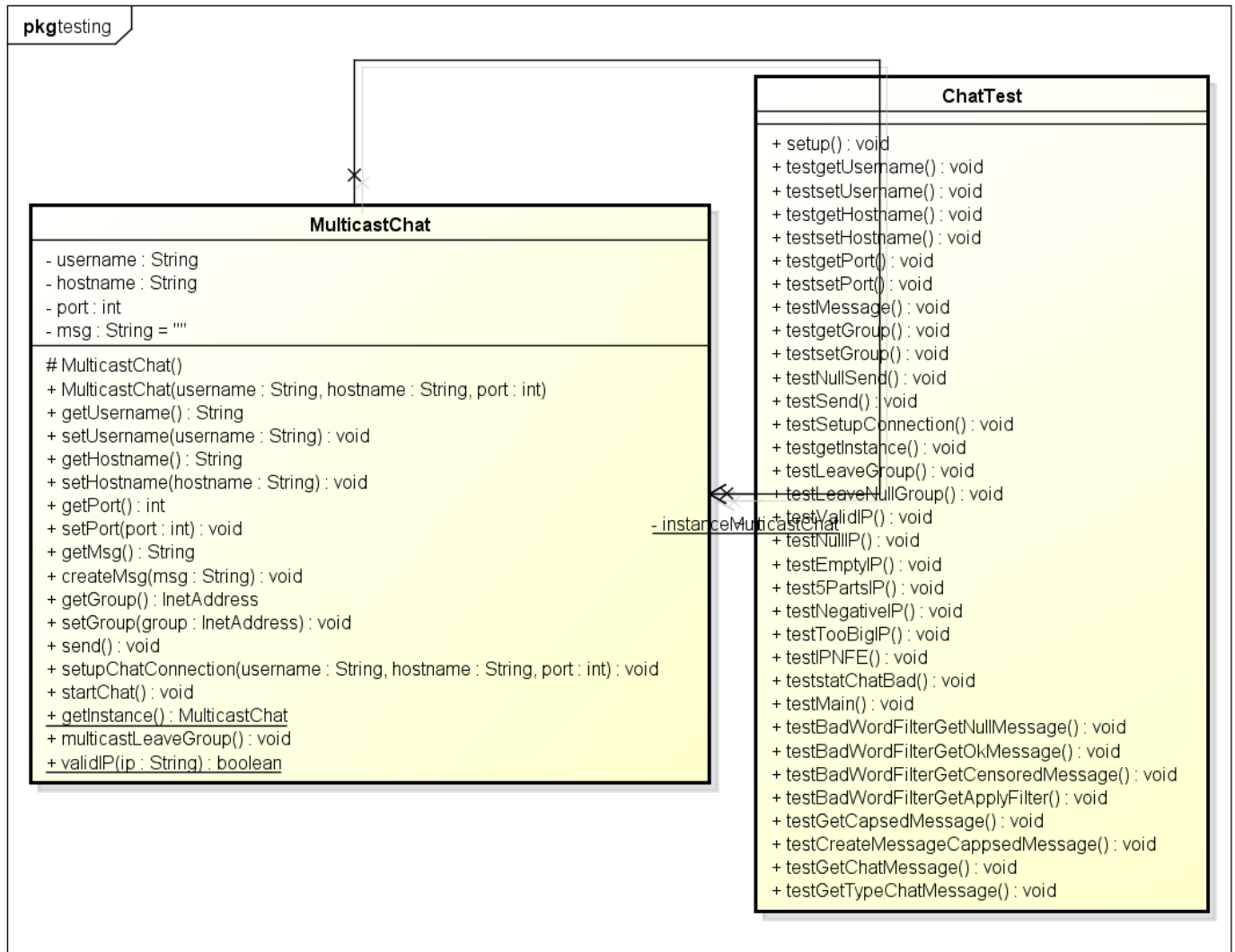




pkgmain

Main

+ main(args : String[]) : void



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.


Element	Coverage	Covered Instru...	Missed Instruct...	Total Instructio...
▲ S04 - Chat für Schwerhörige	 71,4 %	1.223	490	1.713
▲ src	 71,4 %	1.223	490	1.713
▸ stokic_taschner.gui	 44,5 %	311	388	699
▸ stokic_taschner.chat	 81,6 %	387	87	474
▸ stokic_taschner.testing	 97,2 %	517	15	532
▸ stokic_taschner.main	100,0 %	8	0	8


Finished after 0,313 seconds


Runs: 32/32 ❌ Errors: 0 ❌ Failures: 0





▲ stokic_taschner.testing.ChatTest [Runner: JUnit 4] (0,281 s)


 testgetPort (0,000 s)


 testEmptyIP (0,000 s)


 testBadWordFilterGetNullMessage (0,000 s)


 testgetGroup (0,000 s)


 testCreateMessageCappedMessage (0,000 s)


 testIPNFE (0,000 s)


 testNullSend (0,000 s)


 testLeaveNullGroup (0,000 s)


 testMain (0,234 s)


 testSend (0,016 s)


 testBadWordFilterGetCensoredMessage (0,000 s)


 testLeaveGroup (0,000 s)

 teststatChatBad (0,000 s)

 testsetHostname (0,000 s)

 testsetUsername (0,000 s)

 test5PartsIP (0,015 s)

 testnetHostname (0,000 s)

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/>