INF 2 B 2007/2008

M226: Projekt

Netzwerk-Chat erstellen

Emanuel Duss, Arno Galliker, Semir Jahic

13.06.2008

Ein Multicast-Chat

1. Inhaltsverzeichnis

2. Projektantrag

2.1. Projektantrag / Projektauftrag

Projektbezeichnung	Netzwerkchat
Aktuelle Situation, Hintergrund	Momentan verwenden wir zum Versenden von kleinen Nachrichten ein eMail-Programm. Da dies nicht komfortabel ist und dem Mailserver eine Belastung zuträgt, wollen wir dies ändern. Denn
kurze Beschreibung des Projektinhaltes	Wir werden ein kleines Chat-Programm schreiben, dieses Programm kommuniziert über das Netzwerk.
Ziele	Chat-Programm mit grafischer Oberfläche. Versenden von einfachen Text-Nachrichten über eine UDP Verbindung. Die Applikation wird über das Netzwerk laufen. Das Programm soll Objektorientiert implementiert werden. Etwas lernen zur Programmierung in einem Netzwerk (damit auch wir Systemtechniker etwas davon haben). Bei Möglichkeit und Zeit: Wir wollen alle Aktionen und Übermittlungen in einer Datenbank speichern.
Projektumfang / Abgrenzungen	Wir könnten noch eine Bild-, Audio, Video- und Dateiübertragungsfunktion implementieren. Man könnte auch alle Accounts auf einem Server verwalten.
Projektorganisation	Emanuel Duss (Projektleiter) Arno Galliker Semir Jahic
Termine / Meilensteine	2008-05-23: Abgabe Projektantrag 2008-06-06: Dokumentation abgeben
grobe Kostenschätzung	Wir sind arme Schüler und werden nicht bezahlt.
Risiken	Ein Risiko wäre, dass die Applikation nicht im Schulnetz laufen würde, wegen der Sicherheitseinstellungen. Da man aber einen Chat mit Netcat realisieren kann, denken wir, dass es schon funktionieren wird.
Freigabeantrag	Auftragnehmer: Auftraggeber:

3. Projektplanung

3.1. Code-Management

Wir werden versuchen den Code über das Internet zu managen. Dafür kommen folgende Angebote in Frage:

- Google Code
- Sourceforge.org
- Origo.ch
- In der Schule (E)manuell von Hand

Nach langer Diskussion haben wir uns für eine für uns praktischere Methode entschieden: Da nur ein Projektmitglied den Hauptcode macht, tauschen wir den Code nicht über eine Internet-Plattform. Arno schreibt den Grossteil des Codes und er zeigt uns die Ergebnisse. Wir unterstützen ihn durch verschiedene Vorschläge und versuchen zusammen alle Fehler zu lösen.

3.2. Arbeitspakete definieren

Nr.	Beschreibung	T Geschätzt [h]
01	Projektleitung {Kommunikation, Einfordern von Arbeiten, etc.}	2
02	Erstellung eines Prototypen	8
03	GUI-Design erstellen	2
05	Testprotokoll beschreiben	1.5
06	Netzwerkverbindung herstellen	
07	UML-Diagramme zeichnen	6
Total:		**13.5**

3.3. Arbeitspakete auf Ressourcen zuteilen

1.1.1. Emanuel Duss

Nr.	Beschreibung	Geschätzt	Effektiv
1	Projektdokumentation (zusammenfügen)	3	3
2	Infos über Netzwerkverbindungen mit C#, implementieren	4	4
	Programmiertechnische Überlegungen machen	2.5	2.5
3	Projektmanagement	2	2
Total:		**11.5**	**11.5**

1.1.2. Arno Galliker

Nr.	Beschreibung	Geschätzt	Effektiv
1	GUI-Design	3	3
2	Programm-Code schreiben	6	7

Total:	**9**	**10**

1.1.3. Semir Jahic

Nr.	Beschreibung	Geschätzt	Effektiv
1	UML-Diagramme	4	4
2	Programmabläufe planen	2	2
3	Dokumentation	3	3.5
Total:		**9**	**9.5**

4. Vorstudie

4.1. Übermitteln von Text übers Netzwerk

Folgenden Code habe ich auf http://www.wer-weiss-was.de gefunden. Hier wird eine Netzwerkverbindung erstellt. Dieser Code ist zwar in VisualBasic.net. Er half uns jedoch sehr viel weiter:

```
Dim tcpClient As New System.Net.Sockets.TcpClient()
tcpClient.Connect("127.0.0.1", 8000)
Dim networkStream As NetworkStream = tcpClient.GetStream()
Dim sendBytes As [Byte]() = Encoding.ASCII.GetBytes("Is anybody there")
networkStream.Write(sendBytes, 0, sendBytes.Length)
' Read the NetworkStream into a byte buffer.
Dim bytes(tcpClient.ReceiveBufferSize) As Byte
networkStream.Read(bytes, 0, CInt(tcpClient.ReceiveBufferSize))
' Output the data received from the host to the console.
Dim returndata As String = Encoding.ASCII.GetString(bytes)
Console.WriteLine(("Host returned: " + returndata))


Const portNumber As Integer = 8000
Dim tcpListener As New TcpListener(portNumber)
tcpListener.Start()
Console.WriteLine("Waiting for connection...")
Dim tcpClient As TcpClient = tcpListener.AcceptTcpClient()
Console.WriteLine("Connection accepted.")
' Get the stream
Dim networkStream As NetworkStream = tcpClient.GetStream()
' Read the stream into a byte array
Dim bytes(tcpClient.ReceiveBufferSize) As Byte
networkStream.Read(bytes, 0, CInt(tcpClient.ReceiveBufferSize))
' Return the data received from the client to the console.
Dim clientdata As String = Encoding.ASCII.GetString(bytes)
Console.WriteLine(("Client sent: " + clientdata))
Dim responseString As String = "Connected to server."
Dim sendBytes As [Byte]() = Encoding.ASCII.GetBytes(responseString)
networkStream.Write(sendBytes, 0, sendBytes.Length)
Console.WriteLine(("Message Sent /> : " + responseString))
'Any communication with the remote client using the TcpClient can go here.
'Close TcpListener and TcpClient.
tcpClient.Close()
tcpListener.Stop()
Console.WriteLine(e.ToString())
```

Anhand dieses Beispiels sehen wir, wie eine TCP-Verbindung aufgebaut wird.

4.2. Konsolenanwendung oder Windows-Forms

1.1.4. Das Problem

Wir haben eine sehr grosse Problematik: Wir sind unterschiedlicher Meinungen, wie wir das Programm erstellen wollen. Deshalb ist es wichtig, dass wir einen Kompromiss fassen und zusammen ein professionelles Programm erstellen können.

Entweder

- Eine Konsolenanwendung oder
- Eine Windows-Forms Anwendung

1.1.5. Gegenüberstellung

Hier ist eine Gegenüberstellung von den beiden Vorschläge:

	CLI	GUI
Pro	• Schnell • Einfach zu bedienen	• Normal für Windows-Standardbenutzer
Kontra	• Für „normale" Benutzer etwas ungewöhnlich	• Hat nicht so Style… ☺

1.1.6. Genauere Analyse

Was ist einfacher realisierbar?

Grundfrage: Wie kann gleichzeitig empfangen und gesendet werden? Der Bildschirm muss sich aktualisieren und man muss gleichzeitig schreiben können. Dies führt auf der Konsole zu folgendem Problem.

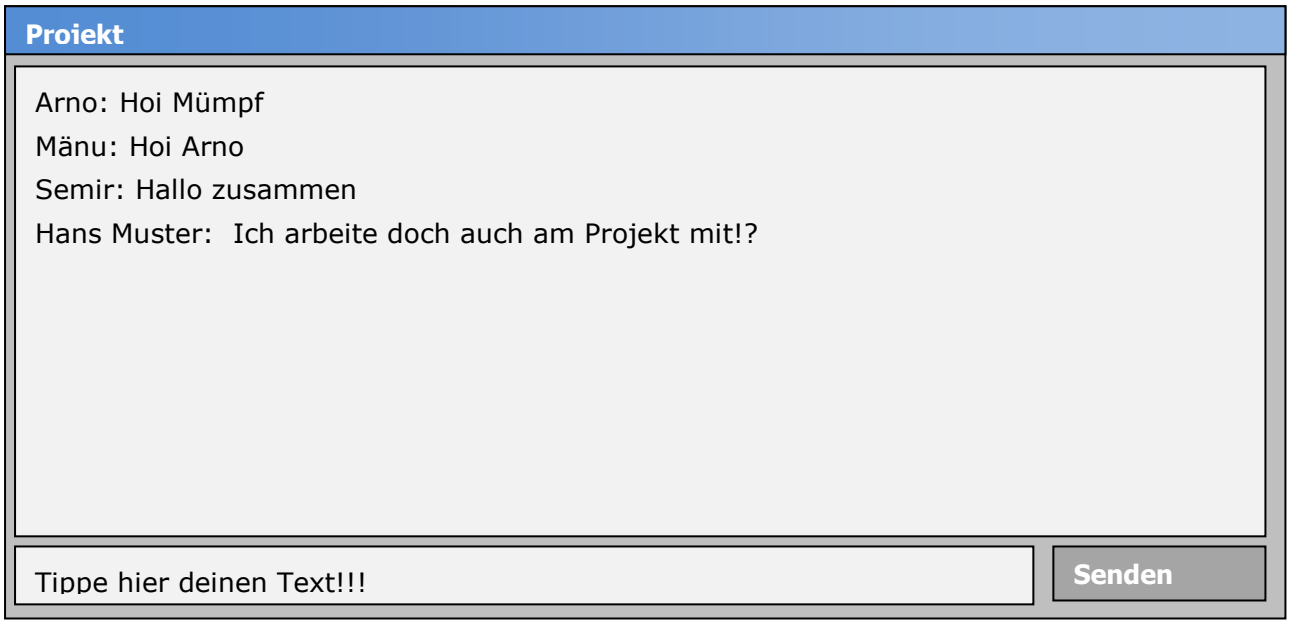
Konsole:

```
Arno: Hoi Mümpf
Mänu: Hoi Arno
Semir: Hallo zusammen.
Hans Muster: Und was ist mit mir???
```

Wie kann ich nun den Text anzeigen lassen und gleichzeitig schreiben? Das ist (für uns) nicht möglich zu realisieren.

Deshalb fällt nun der tolle Konsolenchat aus dem Projekt. Wir werden ein GUI-Projekt erstellen.

Bei einem GUI ist dies Problemlos möglich:



Bei der oberen mehrzelligen Textbox kann man den zu empfangenden Text anzeigen lassen und bei der unteren einzeiligen Textbox kann man den Text schreiben, den man über senden verschickt.

4.3. Wie viele Leute werden am Chat teilnehmen?

Es wäre ja natürlich schön, wenn nicht nur eine öde 2-Mann Verbindung aufgestellt werden, sondern mehrere Leute am Chat mitmachen könnten.

Das Zauberwort heisst: Multicast!

Wir verschicken die Nachricht an eine Multicastadresse. Somit können alle, die den Chat offen haben, mitchatten.

Security: Jeder (!) kann hören, wer was sendet.

Das stört uns ja nicht, da Google sowieso schon alles über uns weiss.

4.4. Verbindungsart

Wir setzen ausserdem auf die verbindungslose UDP-Verbindung. Dabei wird keine Kommunikation mit Handshake ausgehandelt, sondern einfach gesendet. Das ist einfacher zu realisieren, denn UDP orientiert sich nicht an der anderen Seite des Chats, sondern sendet das Signal einfach los, in der Hoffnung, dass es jemand mitkriegt. In unserem konkreten Beispiel, läuft es wie folgt, alle auf dem Port aktiven Teilnehmer, können dem Chat beitreten und ein Signal senden. Ebenfalls ist das mithören möglich.

4.5. Welchen Port

Wir werden den Port verwenden, der vorprogrammiert wurde, das Beste ist, dass wir einen Port benutzen, der über 1000 liegt. Da diese nicht für spezielle Applikationen reserviert sind. Die Wahl auf welchem Port die Unterhaltung stattfinden soll, wird evtl. in einer vom Chatpartner vorgegebenen Textbox mitgegeben.

5. Analyse / Definition

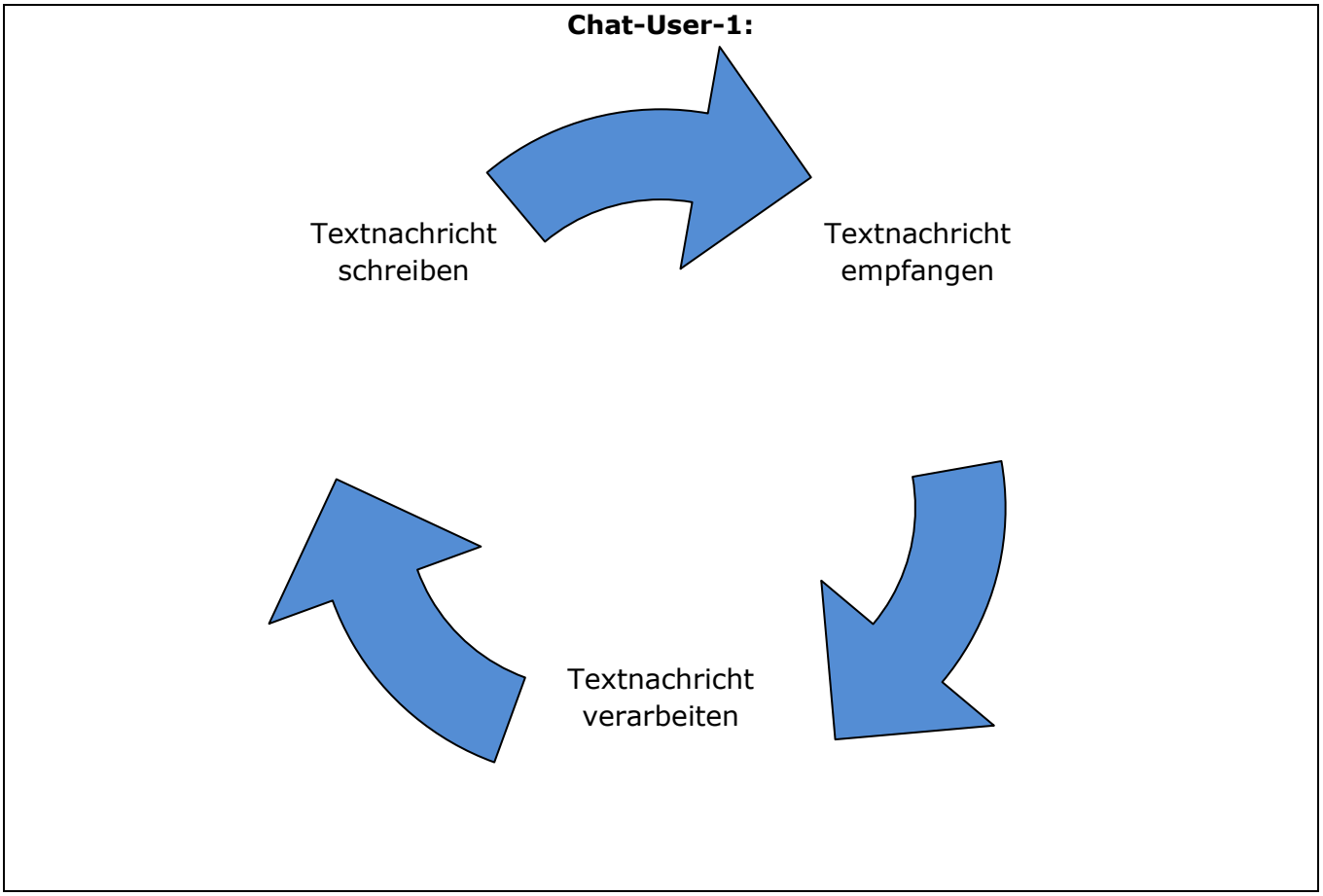
5.1. Anwendungsfallanalyse

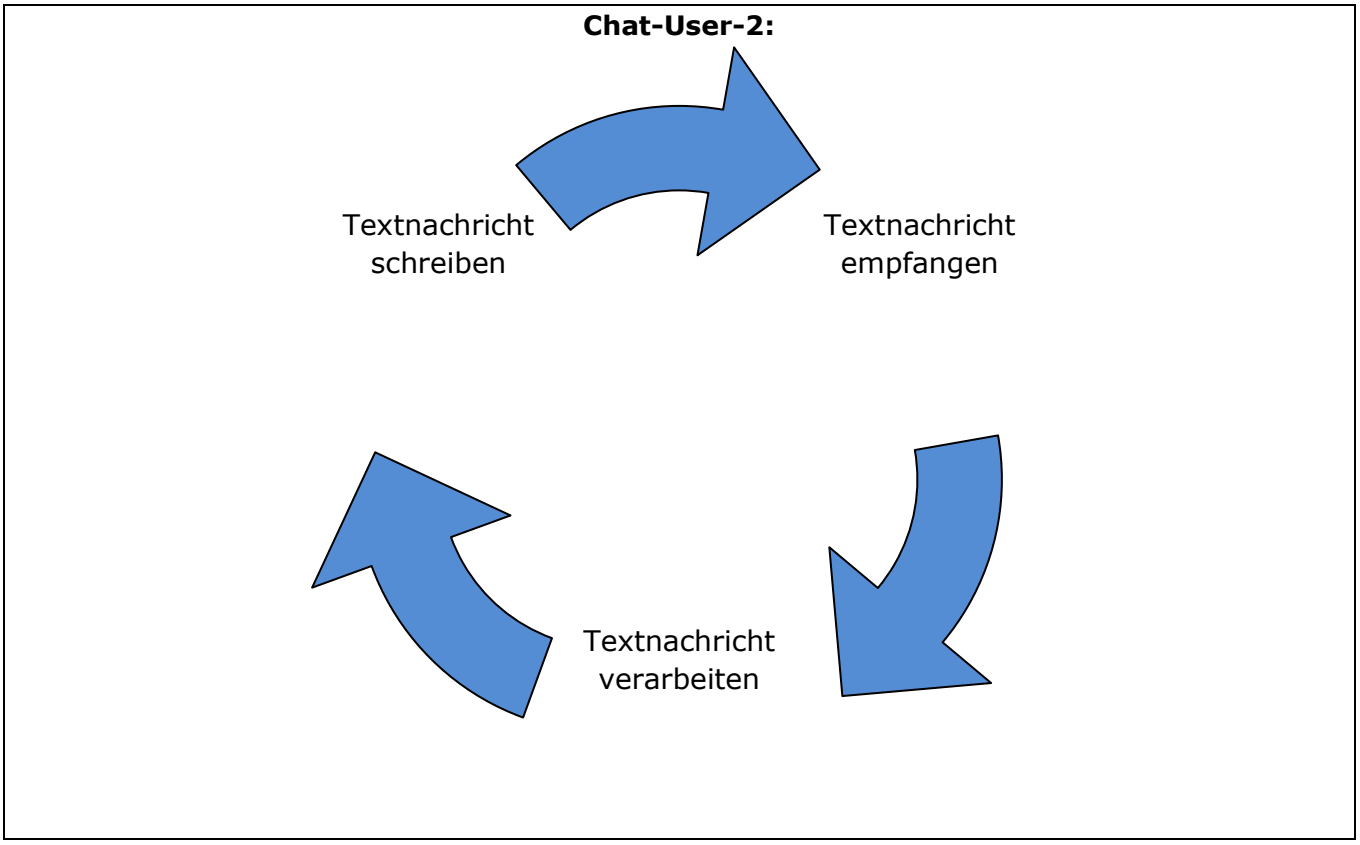
1.1.7. Akteure "Actors" beschreiben

Bei unserem Projekt gibt es nur den Chat-User als Actor. Dieser Actor hat jedoch die Möglichkeit, mit wiederum anderen Actors zu kommunizieren. Somit gibt es nur diese Möglichkeit zur Beschreibung eines Actors.

Chat-User 1	Textnachricht schreiben	• Der User gibt eine Textnachricht in das Eingabefeld ein.
	Textnachricht empfangen	• Der User bekommt eine Antwort auf den gesendeten Port
	Textnachricht lesen	• Die empfangene Textnachricht wird vom User gelesen und im Kopf weiterverarbeitet. Dann beginnt er wieder von Vorne zu schreiben.
Chat-User 2	Textnachricht schreiben	• Der User gibt eine Textnachricht in das Eingabefeld ein.
	Textnachricht empfangen	• Der User bekommt eine Antwort auf den gesendeten Port
	Textnachricht lesen	• Die empfangene Textnachricht wird vom User gelesen und im Kopf weiterverarbeitet. Dann beginnt er wieder von Vorne zu schreiben.

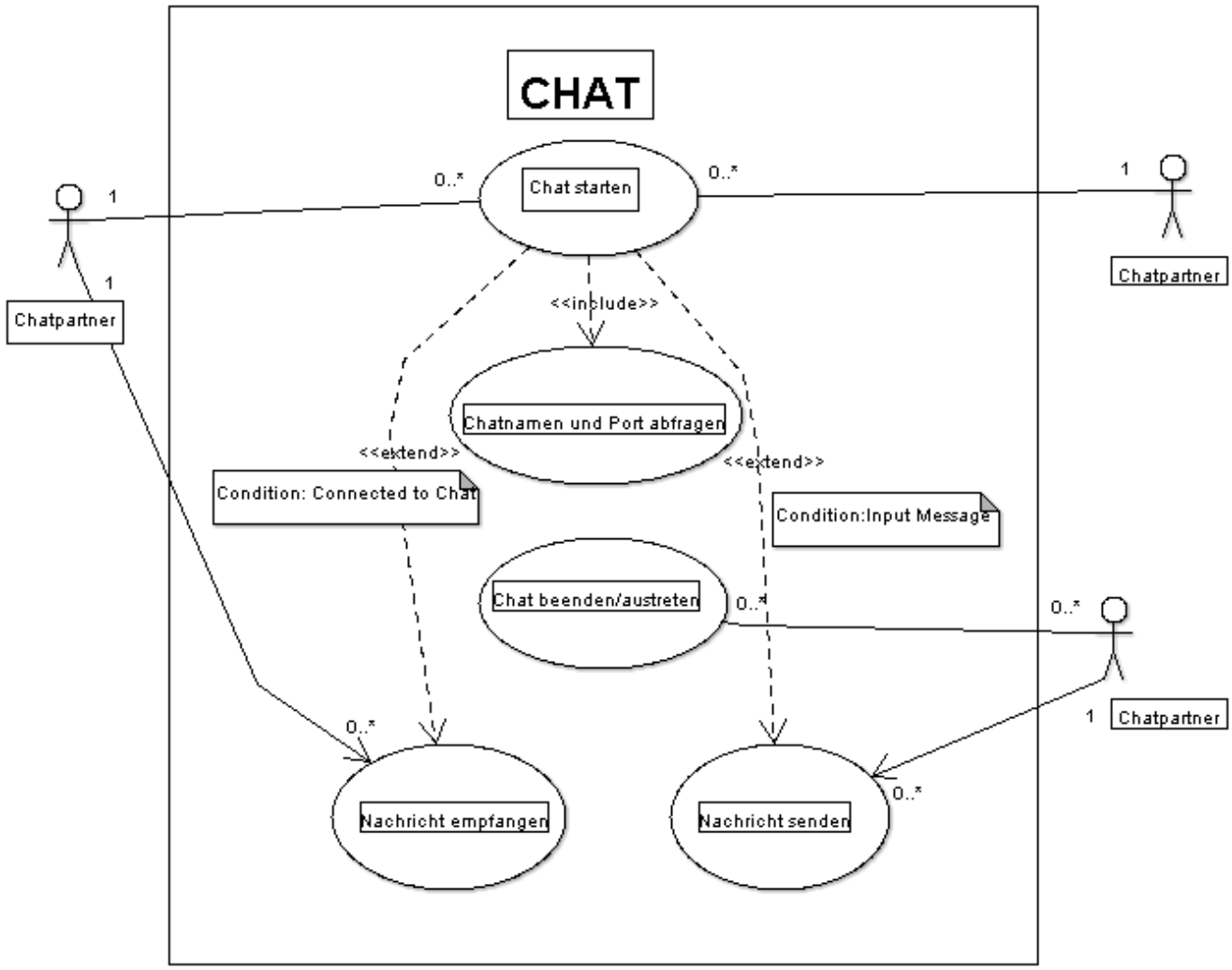
1.1.8. Grafische Darstellung





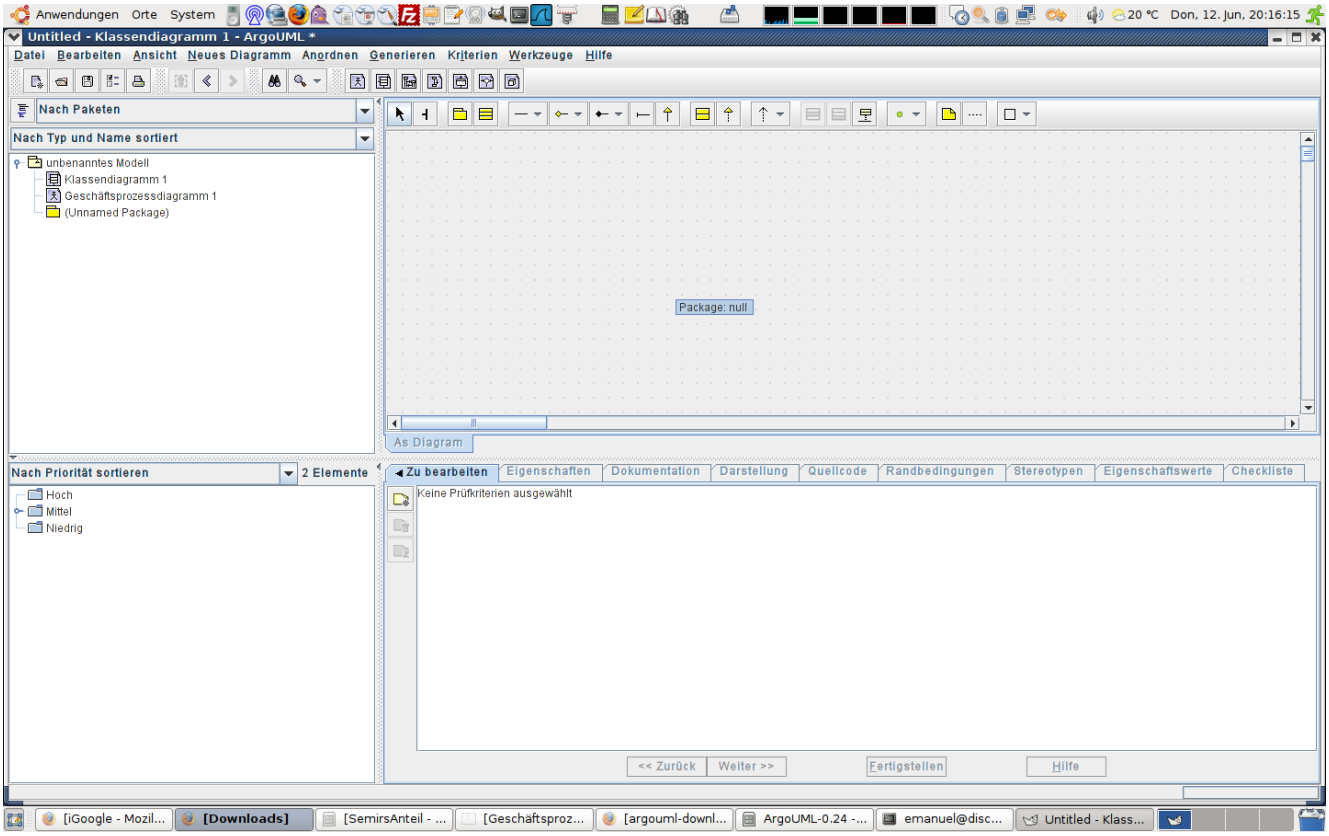
1.1.9. Anwendungsfalldiagramm

{ Erstellen Sie mit einem Visualisierungsprogramm ihrer Wahl (z.B. Enterprise Architect) das Anwendungsfalldiagramm und die entsprechenden Beziehungen. }



1.1.10. Präzise Beschreibung der Anwendungsfälle

Das USE-CASE Anwendungsfalldiagramm habe ich mit ArgoUML designt. Es ist ein frei zugängliches, javabasierendes UML Designprogramm. In diesem UML-Anwendungsfalldiagramm wird beschrieben, was genau abläuft, während der Laufzeit des Programms. Es wird also im Voraus geplant, was das System überhaupt leisten können soll. Das Use-Case-Diagramm beschreibt den Verlauf aus der Sicht der Benutzer, im Jargon werden diese als Akteure bezeichnet. In den sog. Anwendungsfällen, beschreibt man, was genau für Aktionen ausgeführt werden.



1. Der Anwendungsfall „Chat starten". Der Akteur „Chatpartner" kann durch Doppelklick, die implementierte Anwendung zur Erscheinung bringen, der Doppelklick initialisiert das Programm und eröffnet die Möglichkeit zum Partizipieren an einer Konversation. Ein „Chatpartner" kann beliebig viele Instanzen des Programms eröffnen und kann auf verschiedenen Ports, mit diversesten anderen „Chatpartnern" in Kontakt treten. Der Anwendungsfall des „Chat starten", enthält die Abfrage des Ports, auf welchem der Chat aktiv ist, dieser ist für alle Mithörer frei zugänglich. Auch wird dem Chat ein Name gegeben, dieser muss auch eingegeben werden.
2. Der Anwendungsfall „Nachricht senden". Hier tritt ein Chatpartner bewusst in eine Unterhaltung ein, da er an der Kommunikation mit anderen teilnehmen will oder zu einer neuen Konversation aufrufen möchte. Er hat also das Verlangen, einen neuen Chat nun zu nutzen, in dem er beginnt zu kommunizieren. Ebenfalls kann das „Nachricht senden" als Antwort auf eine andere Nachricht sein, diese Art der Kommunikation nennt sich Dialog und ist der primäre Endzweck eines Chats. Durch das Senden von Nachrichten auf der einen Seite, wird als Konsequenz auf der anderen Seite der Unterhaltung eine Botschaft empfangen. So schliesst sich der Kreis des Gesprächs über den Chat.

5.2. Minipflichtenheft (Anforderungen)

- **V** → Verzicht
- **W** → Wunsch-Anforderung
- **M** → Muss-Anforderung

1.1.11. Funktionale Anforderungen

A-Nr	Bezeichnung	Priorität *
1000	Das Programm muss Text ins Netzwerk senden können.	M

1001	Das Programm muss Text, der ins Netzwerk gesendet wurde empfangen und für den User gerecht darstellen können.	M

1.1.12. Datenmengenanforderungen

A-Nr	Bezeichnung	Priorität
1100	Man muss mehrere Tausend Zeichen auf dem Bildschirm anzeigen können.	M

1.1.13. Datenverarbeitunsanforderungen

A-Nr	Bezeichnung	Priorität
1200	Mehrere Benutzer können am Chat teilnehmen	M

1.1.14. Bedienungsanforderungen

A-Nr	Bezeichnung	Priorität
1300	Die ganze Applikation muss per Tastatur (ohne Maus) bedienbar sein.	W
1101	Man muss den Chatverlauf speichern können (XML oder normales Textfile)	V

1.1.15. Umgebungsanforderungen

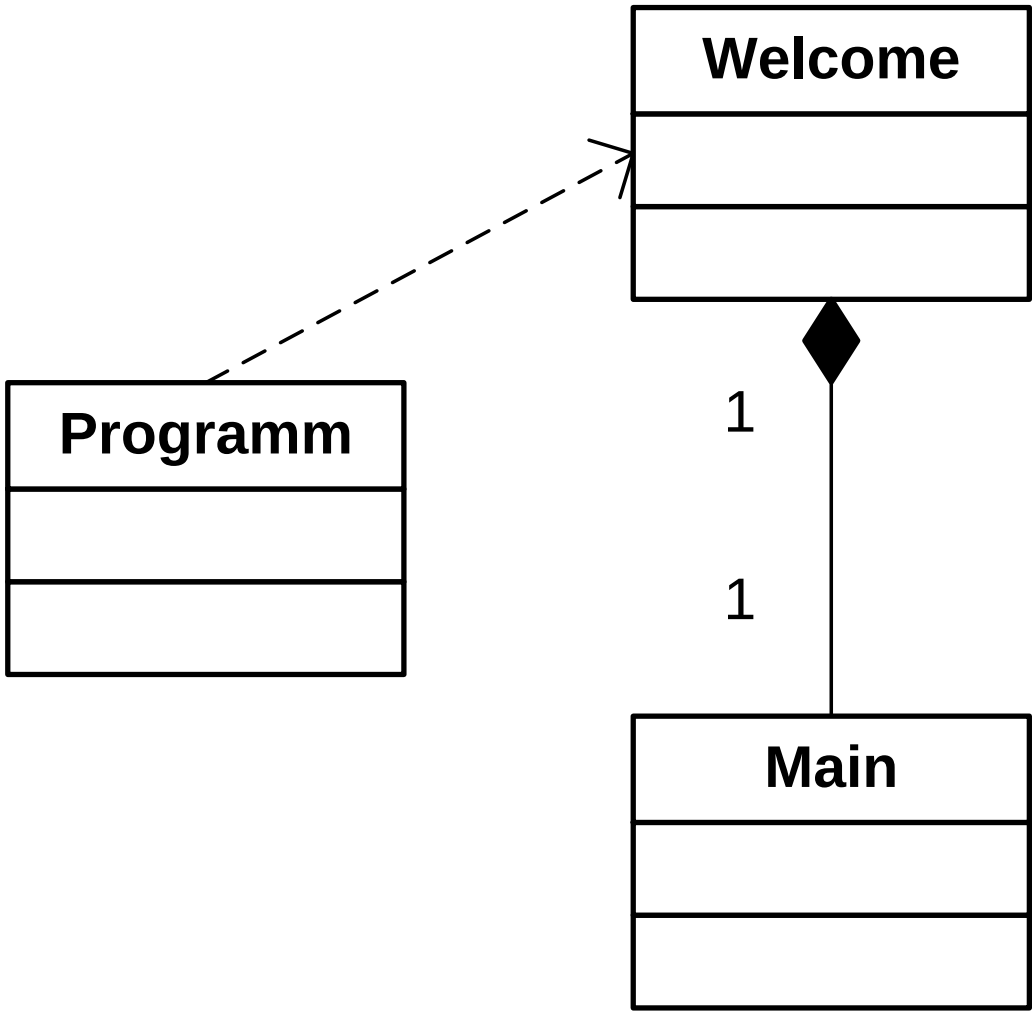
A-Nr	Bezeichnung	Priorität
1400	Die Applikation soll auf Windows XP und auf Vista laufen.	M
1401	Die Applikation soll auf Unix-Sytemen (Linux, BSD, Solaris, etc…) laufen.	V

1.1.16. Finanzielle Anforderungen

A-Nr	Bezeichnung	Priorität
1500	Die Anschaffungs- und Schulungskosten der Applikation dürfen den Bereich von 1.95 sFr nicht übersteigen.	M

5.3. Konzeptuelle Modellierung

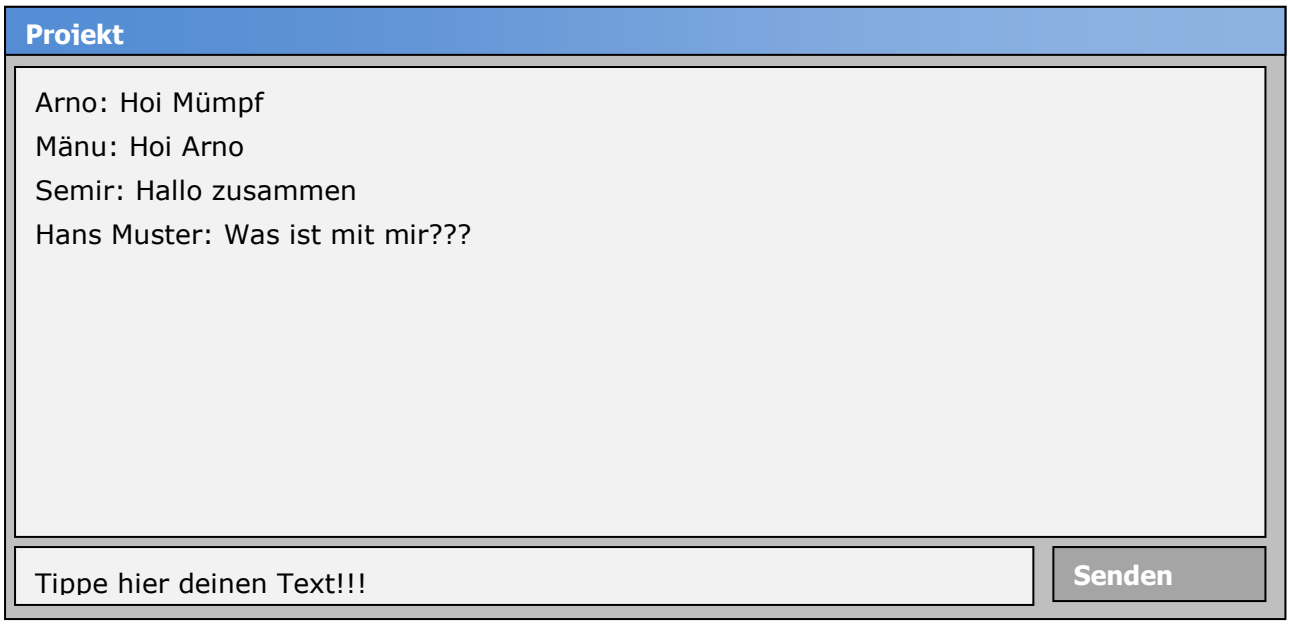
1.1.17. Klassendiagramm mit wenigen Notationselementen



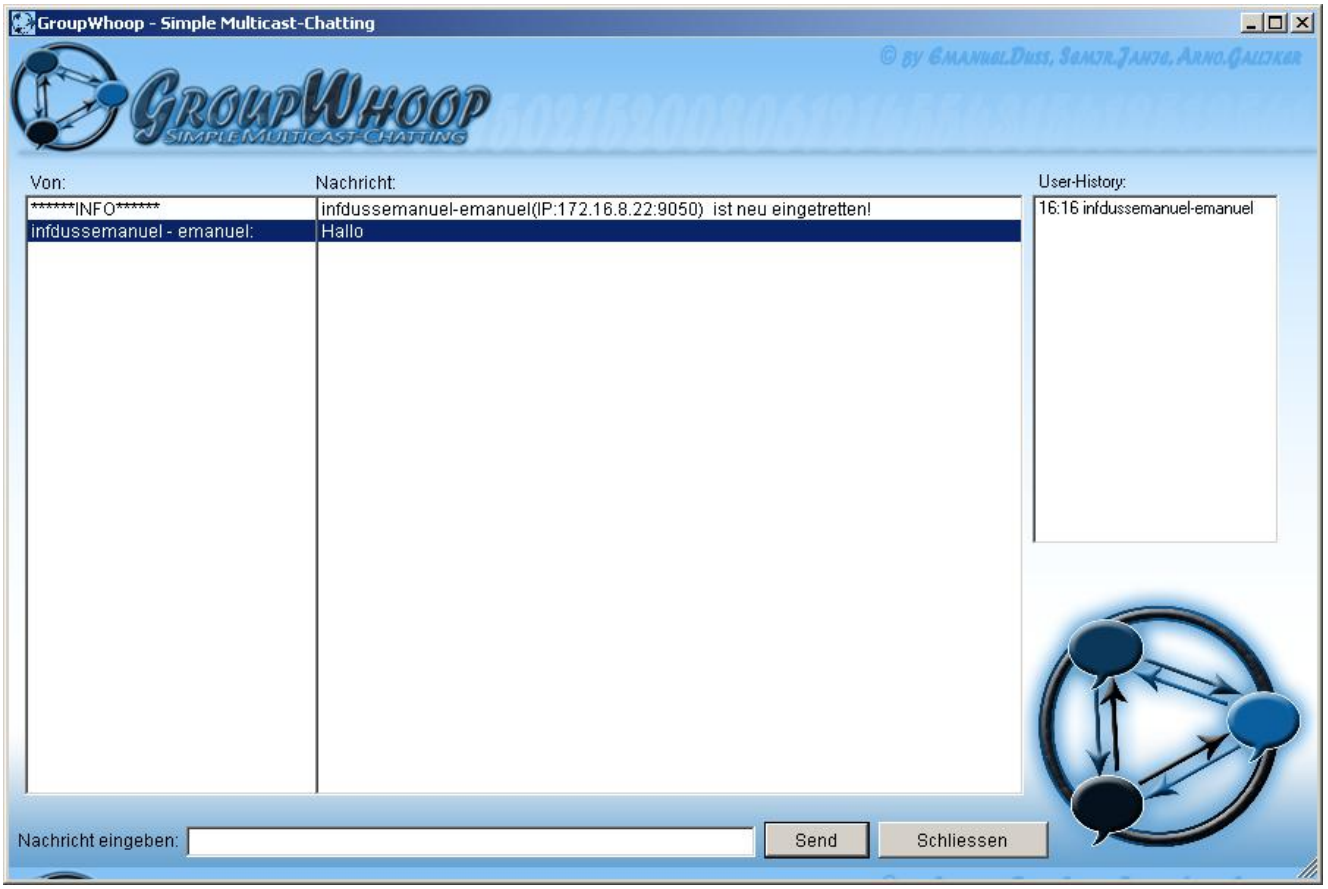
6. Entwurf / Design

6.1. GUI-Design

Folgendes ist das GUI für das Fenster. Es wird nur ein Fenster benötigt.



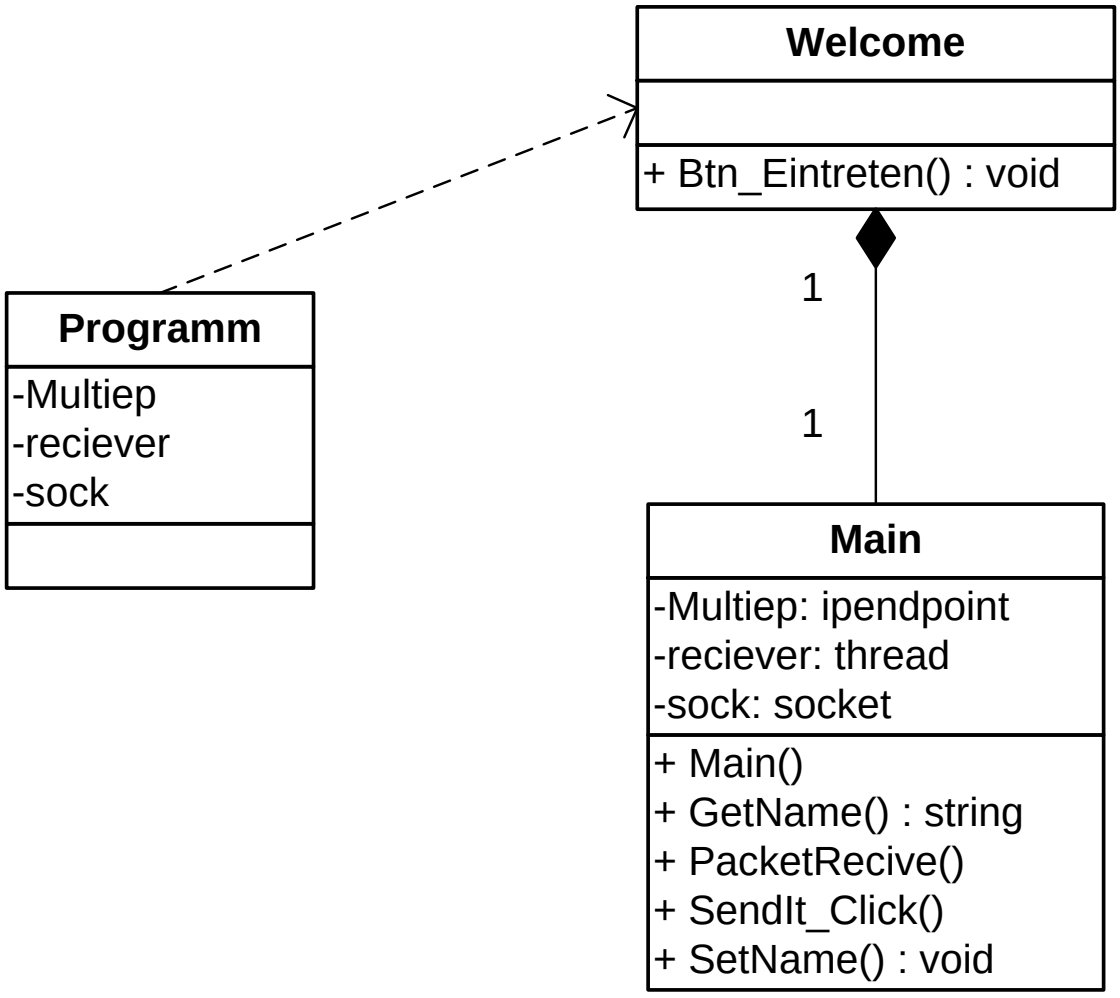
Folgendermassen sieht das definitive GUI aus:



In einer späteren Version könnte man noch ein Optionen-Fenster hinzufügen.

6.2. Applikationsstruktur (Klassendiagramm)

1.1.18. Klassendiagramm detailliert



1.1.19. Beschreibung der Klassen

Folgende Tabelle ist einfach Klasse:

Klasse	Beschreibung
Programm	Ruft die Welcome-Klasse auf. Dabei wird ein Objekt erstellt.
Welcome	Erstellt bei Klick auf den Beitreten-Button ein Objekt der Klasse Main. Dabei wird der abgefragte Name vom User mitgegeben.
Main	Das ist der eigentliche Chat. Hier wird mit der Methode PacketRecieve() werden die Nachrichten empfangen und mit dem SendIt-Button werden sie an die anderen Chatbenutzer verschickt. Beim ersten Aufruf der Main-Klasse wird die Telling-Funktion aufgerufen, welche den Anderen Chatbenutzern mitteilt, dass ein neuer Chatbenutzer eingetreten ist. Beim schliessen des Chats passiert quasi dasselbe. Es wird jedoch mitgeteilt, dass der User den Chat verlassen hat.

7. Test

Wir testen unser Programm mit der Black-Box-Methode.

Black-Box-Test bezeichnet eine Methode des Softwaretests, bei der die Tests ohne Kenntnisse über die innere Funktionsweise des zu testenden Systems entwickelt werden. Er beschränkt sich auf funktionsorientiertes Testen, d. h. für die Ermittlung der Testfälle wird nur die Spezifikation (gewünschte Wirkung), aber nicht die Implementierung des Testobjekts herangezogen. Die genaue Beschaffenheit des Programms wird nicht betrachtet, sondern vielmehr als Black Box behandelt. Nur nach außen sichtbares Verhalten fließt in den Test ein.[1]

7.1. Testfälle

Nr.	Testfall	Erwartet	Erhalten
1	Programm starten	Keine Probleme	Keine Probleme
2	Neuer User findet sofort zurecht	Keine Probleme	Keine Probleme
3	Die Texteingabe erfolgt ohne Probleme	Keine Probleme	Keine Probleme
4	Nach dem drücken der Enter-Taste wird die Nachricht versendet	Keine Probleme	Keine Probleme
5	Nach dem klicken auf die Senden-Taste wird die Nachrcht versendet.	Keine Probleme	Keine Probleme
6	Die gesendete Nachricht wird bei sich selber auch angezeigt.	Keine Probleme	Keine Probleme
7	Ein Gesprächspartner erhält die Nachricht	Keine Probleme	Keine Probleme
8	Alle Gesprächspartner erhalten die Nachricht	Keine Probleme	Keine Probleme
9	Ich erhalte die Nachrcht von einem Chat-Partner	Keine Probleme	Keine Probleme
10	Ich erhalte mehrere Nachrichten, von mehreren Partnern	Keine Probleme	Keine Probleme
11	Es werden „gleichzeitig" abgesendete Nahrichten angezeigt.	Keine Probleme	Keine Probleme
12	Ich kann das Programm erfolgreich und ohne Probleme beenden.	Keine Probleme	Keine Probleme
13	Es können keine leeren Nachrichten versendet werden.	Keine Probleme	Keine Probleme
14	Die Textnachricht wird übersichtlich dargestellt	Keine Probleme	Keine Probleme

[1] Quelle zum kleinen Text: Wikipedia (http://de.wikipedia.org/wiki/Black-Box-Test)

Nr.	Testfall	Erwartet	Erhalten
15	Auch wenn man viel auf einmal schreibt gibt es keine Probleme	Keine Probleme	Keine Probleme
16	Mit Wine unter Linux ausführbahr	Keine Probleme	Keine Probleme

7.2. Fazit zu den Tests

Das Programm hat alle Tests bestanden.

Es ist auf jeden Fall reif für die Markteinführung. Markteinführung hört sich ein bisschen kapitalistisch ein. Wir wollen jedoch kein Geld dafür verlangen, da wir Freie Software (nicht im Sinn von Frei-Bier, sondern im Sinn von Freiheit) lieben! OpenSource heisst das Stichwort mit der GPL-Lizenz!

Wir sind sehr froh darüber, dass unsere Ziele bei weitem erfüllt wurden. Das Programm hat alle Tests problemlos bestanden und wir haben ein grosses Erfolgserlebnis genossen.

8. Reflexion

8.1. Persönliche Reflexion

1.1.20. Emanuel Duss

Das Projekt war gut. Es war nicht so super, wie ich es mir zuerst vorgestellt hatte. Ich habe gehofft, dass wir mehr Zeit haben. Wir konnten nämlich nicht alles machen, was uns in der Schule beigebracht hätte sein müssen. Es war ein riesen Stress. Trotzdem ist unser Programm am Ende erfolgreich gekommen. Die Zusammenarbeit mit Arno und Semir war gut, wir haben die Aufgaben aufgeteilt und jeder versucht sein Bestes zu geben. Ich wollte zwar Programm in der Konsole schreiben, aber nun haben wir das Chatprogramm mit GUI. Der Netzwerkchat ist simpel, aber wir haben auch noch nicht so grosse Fähigkeiten in C#. Ich hätte lieber mehr Zeit gehabt, zum testen und alles. So wie wir stressen mussten war es ganz schlecht, wir mussten sogar zuerst dokumentieren, bevor wir dann richtig programmiert haben. Der Spass war da, jedoch haben wir viele Probleme gehabt am Anfang. Wir konnten uns nicht einigen und das Projekt war sehr kompliziert. Niemand hat gewusst, was wir machen müssen am Anfang. Aber schlussendlich war es ein Erfolg, wir haben gut zusammengearbeitet und das tolle Projekt gelang super. Nun habe ich ein paar Kenntnisse in C#, die ich vielleicht nutzen kann im Beruf.

1.1.21. Arno Galliker

Mein Fazit zu unserem Projekt fällt positiv aus, obwohl wir nicht genügend Zeit hatten, um alle Ideen zu realisieren. Ich konnte gut meine Kenntnisse vertiefen und habe viel dazugelernt. Grundsätzlich war ich sehr motiviert, ich habe viel am Code gearbeitet und hätte gerne den Debugprozess genauer betrachtet. Der Netzwerkchat war sehr interessant und ich – als Systemtechniker – konnte auch noch unsere Netzwerkkenntnisse ein Stück weit im Projekt anwenden. Nicht so gut fand ich, dass die Zeitplanung so eng war. Ich hätte viel mehr Zeit investiert, sonst ist es nicht möglich, dass Programm vollständig und sauber zu programmieren. Das Vergnügen mit Emanuel und Semir war gross, wir hatten die Möglichkeit unsere Kenntnisse zu verschmelzen und gemeinsam ein gutes Projekt zu realisieren. Ein solches Projekt zu machen ist sehr zeitaufwendig, aber ich nehme mir auch gerne Zeit dafür. Ich fand es schön, wir waren interessiert dabei und wenn etwas gelang war die Euphorie wunderbar. Nun bin ich über den gesamten Verlauf zufrieden. Ich bin jetzt zwar noch kein C#-Profi aber ich besitze gute Grundkenntnisse und verstehe Codes, wenn ich diese ansehe.

1.1.22. Semir Jahic

Ich fand das Projekt sehr interessant, die Fähigkeit die ich in der Schule erlernt habe, konnte ich gut umsetzen. Ich fand es super, dass wir objektorientiert arbeiten durften. Jedoch war die Zeit der entscheidende Faktor, welcher unsere Freude getrübt hat. Denn in zwei Wochen ist es äusserst schwer ein solches Projekt zu erarbeiten. Der Netzwerkchat war ein gutes Programm, wir haben Kenntnisse über Netzwerk und Programmierung in C# zusammengefügt und sie im Programm vereint. Nun können wir erfolgreich miteinander chatten im internen Netzwerk. Ich meine, wir alle haben schon in Chatrooms miteinander geredet oder chatten mit anderen im ICQ oder MSN. Jedoch habe ich nun hintergründiges Wissen über die Funktion solcher Programme, die die Kommunikation im Netzwerk ermöglichen. Rückblickend kann ich sagen, wir haben in der Schule viel lernen können. In der Praxis werden wir kaum jemals etwas von dem verwenden, jedoch habe ich persönlich nun eine Vorstellung, was Begriffe wie: OOP, UML und .NET heissen. So kann ich auch als Systemtechniker die Programmierer verstehen. Für die Zukunft ist es mir wichtig, alle Aspekte der Informatik, mindestens ansatzweise zu kennen.

1.1.23. Persönliche Reflexion von Hans Mustermann

Ich war zwar nicht so integriert in die ganze Projektarbeit. Aber ich habe meine Kenntnisse sehr vertiefen können. Von Arno, Emanuel und Semir konnte ich sehr viel lernen, diese haben bereits viel gelernt und den Grossteil erarbeitet. Deshalb bin ich froh musste ich nicht viel machen. Denn eigentlich habe ich Angst, dass mich niemand ernst nimmt. Ich habe eigentlich nur hier in der Reflexion meinen Beitrag leisten können. Ich hoffe ich werde keine langfristigen, psychologischen Traumata davontragen. Der ständige Druck der gegenwärtigen Gesellschaft zerstört meine Kreativität und zwängt alle meine Inspiration in logische, kleine Einheiten ohne Sinn. Ich fühle mich indoktriniert und kann nicht ausbrechen, denn niemand beachtet mich. Nun, ich habe keine grossen Kenntnisse in C# aber ich hoffe dass die Zukunft mir ein Licht schenkt, sonst muss ich selber die Verantwortung in die Hand nehmen.

8.2. Gruppenreflexion

Wir hatten ein grosse Aufgabe gemeistert und sind nun glücklich darüber, dass wir ein Projekt gemeinsam verwirklicht haben. Der Zeitdruck lastete stets auf uns. Die Möglichkeiten alle unsere Kenntnisse frei umzusetzen hatten wir keineswegs. Trotzdem aber sind wir froh darum, eine Chance erhalten zu haben, in der wir Teile des erarbeiteten Stoffes in einem freien Projekt umsetzen konnten. Im Konsens haben wir die Aufgaben schlussendlich erfolgreich gemeistert. Die Idee eines Netzwerkchats war zu Beginn eine scheinbar unmögliche Aufgabe. Mit der Zeit erkannten wir, wenn wir die Aufgaben gut aufteilen und fleissig arbeiten, das Projekt zum Erfolg führt. Der Chat funktioniert besser als erwartet, es kann nicht nur eine Benutzer zu Benutzer Konversation erfolgen, sondern es können n-Chatmitglieder beitreten. Dies hat ein gruppenweites Erfolgserlebnis ausgelöst. Wir werden das Modul mit einer guten Stimmung und viel gelerntem verlassen dürfen.

Folgende Tabelle mussten wir erstellen, was ich (Emanuel) falsch finde. Jeder von unserer Gruppe hat Vollgas gegeben. Wir haben die Arbeiten gleichmässig aufgeteilt. Wir sind nur zusammen ans Ziel gekommen, weil alle hart mitgearbeitet hatten. (Hans Muster war zwar ein wenig mühsam…)

Deshalb sieht unsere Rangliste folgendermassen aus:

	Quantitative Rangliste (1. = hat am meisten gemacht)	**Komplexitäts- Rangliste** (1. = hat die komplexesten Aufgaben gemacht)
1	Arno Galliker, Semir Jahic & Emanuel Duss	Emanuel Duss, Semir Jahic & Arno Galliker
4.	Hans Muster	Hans Muster

9. Anhang

9.1. Systemvoraussetzungen

- Windows XP oder neuer
- .NET-Framework
- Stehendes Netzwerk

9.2. Anleitung zum eigenen kompilieren

VisualStudio 2005 stürzt ab, wenn wir das Programm direkt mit F5 aus der Entwicklungsumgebung kompilieren und gleich starten. Wir müssen das Programm kompilieren und danach im Ordner Release oder Debug (je nach Modus) per Hand doppelklicken.

Das ist ein Problem mit den Threads. Das war die Aufgabe von Hans Muster. Er hat seine Arbeit wirklich sehr schlecht gemacht.

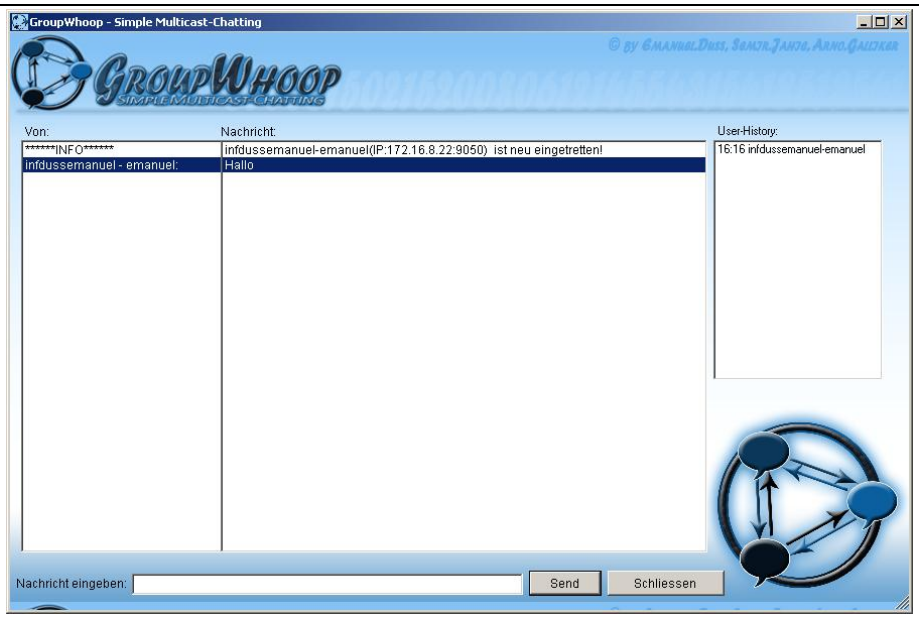
9.3. Installationsanleitung

Das Programm ist portabel und muss deshalb nicht installiert werden. Man kann direkt die EXE-Datei starten.

9.4. Bedienungsanleitung

Das Programm ist eigentlich selbsterklärend:

Programm starten	Man klickt auf das EXE-File.
Namen Eingeben	Man gibt den Usernamen ein: 
Textnachricht schreiben	Man schreibt in die einzeilige Textbox.

	
Textnachricht absenden Variante 1	Man drückt Enter zum Versenden der Nachrichten.
Textnachricht absenden Variante 2	Oder man drückt die Senden-Taste zum Versenden.
Textnachricht empfangen	So intelligent ist das Programm: Hierfür muss man gar nichts machen. Das Programm empfängt die Textnachricht automatisch.
Textnachricht lesen	In der oberen ListBox sieht man den Chatverlauf.

9.5. Sourcecode

Projektverzeichnis.zip

9.6. Ausführbare EXE-Datei

FinalGroupWhoop.exe

10. Anhang: Programmcode

10.1. Programm.cs

```
/*
 * Autoren: Emanuel Duss, Arno Galliker und Semir Jahic
 * M226 Abschlussarbeit "GroopWhoop"
 *
 */

using System;
using System.Collections.Generic;
using System.Windows.Forms;

namespace main
{
    static class Program
    {
        [STAThread]
        static void Main()
        {
            // Hier wird das Programm gestartet.

            // Fenster wird angezeigt
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);

            // Ein Objekt der Klasse Welcome wird initialisiert
            Application.Run(new Welcome());
        }
    }
}
```

10.2. Welcome.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace main
{
    public partial class Welcome : Form
    {

        public Welcome()
        {
            InitializeComponent();
        }

        public void Welcome_Load(object sender, EventArgs e)
        {
        }
```

```
        public void button1_Click(object sender, EventArgs e)
        {
            if (txt_name.Text == "")
            {
                MessageBox.Show("Bitte Namen eingeben!", "Keine Eingabe");
            }
            // Wir fragen nach dem Namen vom Benutzer
            // Name wird dem Konstruktor mitgegeben
            else
            {
                main chater = new main(txt_name.Text);

                // Wir verstecken das Willkommensfenster
                this.Hide();

                // Wir öffnen den Chat-Dialog
                chater.ShowDialog();

                // Das Fenster wird geschlossen (erst wenn die Methode
ShowDialog() abgeschlossen ist)
                this.Close();

            }

        }

    }
}
```

10.3. Main.cs

```
using System;
using System.Drawing;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading;
using System.Windows.Forms;

namespace main
{
    public partial class main : Form
    {
        // Folgende Angaben brauchts um eine Verbindung herzustellen
        Socket sock;
        Thread receiver;
        IPEndPoint multiep = new IPEndPoint(IPAddress.Parse("224.100.0.1"), 9050);

        // Name vom Benutzer
        private string name1;

        // Der name wird gesetzt
        public void setName(string value)
        {
            name1 = value;
        }

        // Der Name kann geholt werden
        public string getName()
```

```
        {
            return name1;
        }

        // Konstruktor
        public main(string value)
        {
            // der Name wird geestzt
            setName(value);

            // Diese Zeile ist nötig, damit das Programm funktioneirt
            InitializeComponent();

            //Es wird eine IP-Verbindung hergestellt
            // UDP-Verbindung
            sock = new Socket(AddressFamily.InterNetwork, SocketType.Dgram,
ProtocolType.Udp);
            // Auf Port 9050
            IPEndPoint iep = new IPEndPoint(IPAddress.Any, 9050);
            sock.Bind(iep);
            // Fehler werden abgefangen mit Try-Catch (z.B. bei Verbindungsfehler:
Keine Netzwerkverbindung)
            try
            {
                // Die oben eingestellten Optionen werden verwendet und gesetzt.
                sock.SetSocketOption(SocketOptionLevel.IP,
SocketOptionName.AddMembership, new
MulticastOption(IPAddress.Parse("224.100.0.1")));
            }
            catch
            {
                // Ausgabe bei Fehlern
                MessageBox.Show("Fehler bei der Verbindungsherstellung.
Netzwerkverbindung überprüfen!");
                // Schliesst es beim Fehelrfall
                this.Close();
            }

            // Pakete sollen empfangen werden
            receiver = new Thread(new ThreadStart(packetReceive));
            // Dieser "Empfänger" arbeitet im Hintergrund
            receiver.IsBackground = true;
            // Nun wird er gestartet.
            receiver.Start();
        }

        // Funktion um Pakete zu empfangen
        void packetReceive()
        {
            // Variabeln definieren
            EndPoint ep = (EndPoint)multiep;
            byte[] data = new byte[1024];
            byte[] chatername = new byte[1024];
            byte[] closemessage = new byte[1024];
            string stringData;
            string stringData2;

            int recv1;
            int recv2;

            // Wichtige Methode: Telling zeigt, dass ein neuer Benutzer online
istr
```

```
            // Wird ausgeführt, BEVOR man online kommt und schreiben kann um den
anderen mitzuteilen, dass ein neuer User online ist
            telling();

            while (true)
            {
                    recv1 = sock.ReceiveFrom(chatername, ref ep);
                    recv2 = sock.ReceiveFrom(data, ref ep);

                    // Hier werden die verschiedenen Pakete in die Stringvariabeln
eingelesen
                    stringData = Encoding.ASCII.GetString(chatername, 0, recv1);
                    stringData2 = Encoding.ASCII.GetString(data, 0, recv2);


                    // Wenn die Nachricht "ich bin neu" entspricht, wird die Info
ausgegeben, das ein neuer Chatter eingetreten ist
                    if (stringData2 == "ich bin neu")
                    {
                        string minute = Convert.ToString(DateTime.Now.Minute);
                        string hour = Convert.ToString(DateTime.Now.Hour);

                        string time = hour + ":" + minute;
                        namerow.Items.Add("******INFO******");
                        results.Items.Add(stringData + "(IP:" + ep.ToString() + ")
" + " ist neu eingetretten!");
                        whowasonline.Items.Add(time + " " + stringData);
                    }
                    // Sonst wird die Nachricht und der Chattername normal in die
2 Listboxen geschrieben.
                    else
                    {
                        namerow.Items.Add(stringData);
                        results.Items.Add(stringData2);

                        // letztes Item selecten, damit automatisch gescrollt wird
                        results.SelectedIndex = results.Items.Count - 1;
                        namerow.SelectedIndex = namerow.Items.Count - 1;
                    }

            }
        }

        // Klick-Event vom Schliessen-Button
        private void exit_Click(object sender, EventArgs e)
        {
            // Vor dem Schliessen eine Nachricht Mitchatter senden: Ich habe den
Chat verlassen
            byte[] closername = Encoding.ASCII.GetBytes("INFO");
            byte[] closemessage = Encoding.ASCII.GetBytes(getName() + " hat gerade
den Chat verlassen!");
            sock.SendTo(closername, SocketFlags.None, multiep);
            sock.SendTo(closemessage, SocketFlags.None, multiep);

            // Fenster schliessen
            Close();
        }



        void telling()
        {
```

```
            // Wenn der Chatter neu hinzukommt, sendet er dank der Telling-
Funktion die Nachricht: "ich bin neu"
            // Dadurch wird den anderen mitgeteilt, dass ein neuer Benutzer
hinzugekommen ist
            byte[] remotechattername =
Encoding.ASCII.GetBytes(System.Environment.UserName + "-" + getName());
            byte[] message = Encoding.ASCII.GetBytes("ich bin neu");

            sock.SendTo(remotechattername, SocketFlags.None, multiep);
            sock.SendTo(message, SocketFlags.None, multiep);

        }

        // Send-Event: Hier wird die Nachricht gesendet
        private void sendit_Click(object sender, EventArgs e)
        {
            if (newText.Text == "")
            {
                MessageBox.Show("Keine Nachricht eingeben!", "Keine Eingabe");
            }

            else
            {
                byte[] remotechattername =
Encoding.ASCII.GetBytes(System.Environment.UserName + " - " + getName() + ":");
                byte[] message = Encoding.ASCII.GetBytes(newText.Text);
                // Die Textbox wird nach dem Senden wieder gelöscht
                newText.Clear();
                sock.SendTo(remotechattername, SocketFlags.None, multiep);
                sock.SendTo(message, SocketFlags.None, multiep);
            }
        }

    }




}
```