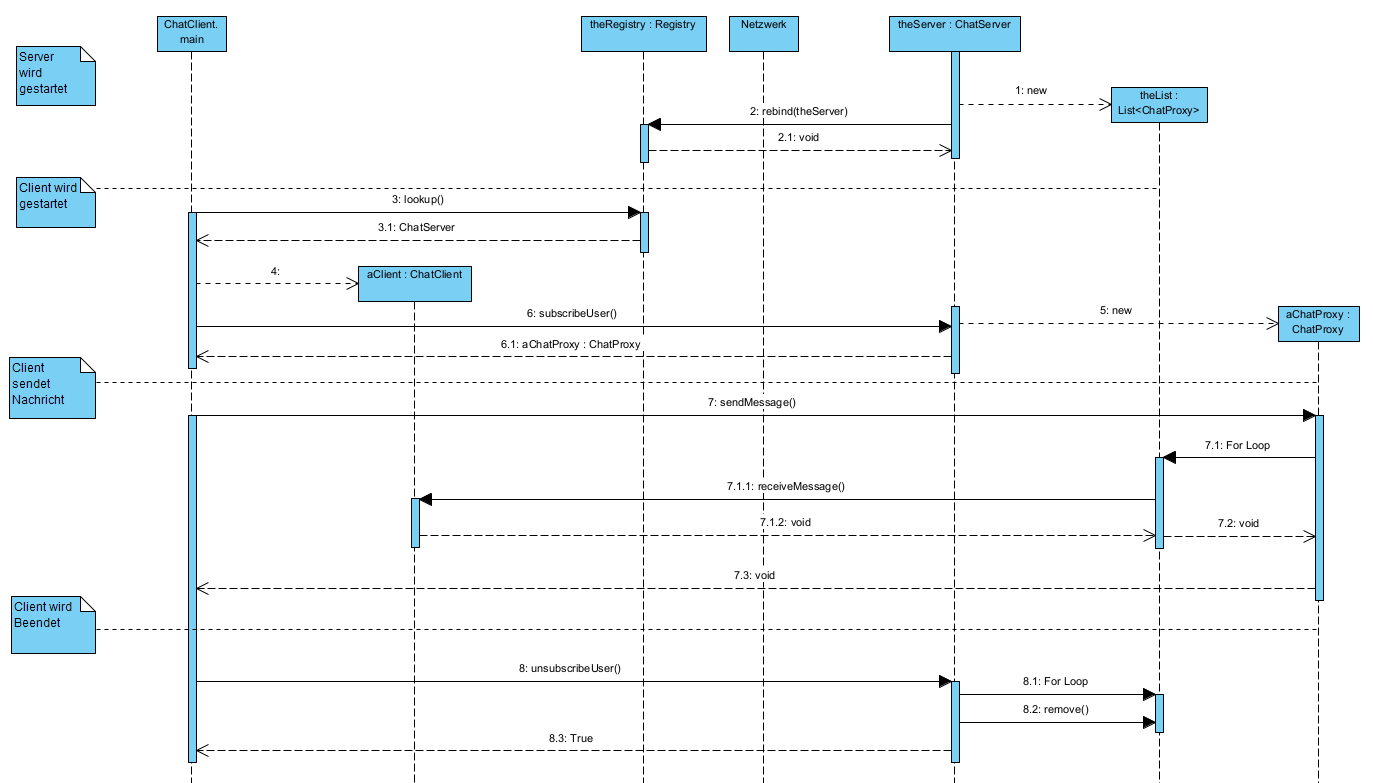
*Verteilte Systeme im Sommersemester 2021*

Steffen Herweg, Matr. Nr. 873475

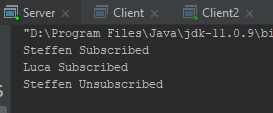
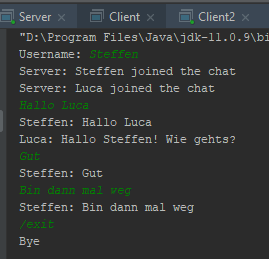
Luca Fabio Kock, Matr. Nr. 879534 Osnabrück, 30.05.2021

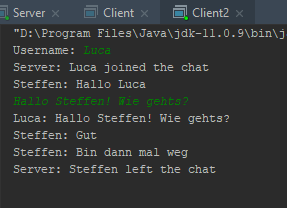
# Aufgabenblatt 6

**Sequenzdiagramm:**

**Tests:**

Funktionierender Chat:





Wenn der Client zu einer falschen Server IP connecten will:



**ClientProxyImpl.java:**

public class ClientProxyImpl extends UnicastRemoteObject implements ClientProxy {  
  
 private ClientProxyImpl() throws RemoteException {  
 super();  
 }  
  
 public static void main(String[] args) {  
 try {  
 Registry registry;  
 String host;  
 if (args.length < 2) {  
 host = "localhost";  
 registry = LocateRegistry.*getRegistry*(Main.*PORT*);  
 } else {  
 host = args[1];  
 registry = LocateRegistry.*getRegistry*(host, Main.*PORT*);  
 }  
  
 try {  
 ChatServer chatServer = (ChatServer) registry.lookup(Main.*NAME*);  
 *mainLoop*(chatServer);  
 } catch (UnknownHostException e) {  
 System.*err*.printf("Unknown Host: %s\n", host);  
 System.*exit*(1);  
 } catch (NotBoundException e){  
 System.*err*.printf("Service not bound on Host %s\n", host);  
 System.*exit*(1);  
 } catch (ConnectException e){  
 System.*err*.printf("Could not Connect to Host: %s \n",e.getCause());  
 System.*exit*(1);  
 }  
  
 }catch (RemoteException e){  
 e.printStackTrace();  
 System.*exit*(1);  
 }  
 }  
  
 private static void mainLoop(ChatServer chatServer) {  
  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.print("Username: ");  
 String username = scanner.nextLine();  
  
 try {  
 ChatProxy chatProxy = chatServer.subscribeUser(username, new ClientProxyImpl());  
  
 while (true) {  
 String input = scanner.nextLine();  
 if (input.equals("/exit")) {  
 chatServer.unsubscribeUser(username);  
 System.*out*.println("Bye");  
 System.*exit*(0);  
 }  
 chatProxy.sendMessage(input);  
 }  
  
 } catch (RemoteException e) {  
 System.*err*.println("Could not get ChatProxy");  
 System.*exit*(1);  
 }  
 }  
  
 @Override  
 public void receiveMessage(String username, String message) throws RemoteException {  
 System.*out*.printf("%s: %s\n", username, message);  
 }  
}

**Main.java:**

package chatsystem;  
import chatsystem.client.ClientProxyImpl;  
import chatsystem.server.ChatServerImpl;  
import java.rmi.registry.Registry;  
  
public class Main {  
  
 public static final int *PORT* = Registry.*REGISTRY\_PORT*;  
 public static final String *NAME* = "ChatServer";  
  
 public static void main(String[] args) {  
 if(args.length < 1){  
 *printUsage*();  
 System.*exit*(1);  
 }  
 switch (args[0]){  
 case "client":  
 ClientProxyImpl.*main*(args);  
 break;  
 case "server":  
 ChatServerImpl.*main*(args);  
 break;  
 default:  
 *printUsage*();  
 }  
  
 }  
  
 private static void printUsage(){  
 System.*err*.println("Usage:");  
 System.*err*.println("chatsystem client [hostaddress]");  
 System.*err*.println("chatsystem server");  
 }  
}

**ChatServerImpl.java:**

public class ChatServerImpl extends UnicastRemoteObject implements ChatServer {  
  
 private class ChatProxyImpl extends UnicastRemoteObject implements ChatProxy{  
  
 private String username;  
  
 ChatProxyImpl(String username) throws RemoteException {  
 this.username = username;  
 }  
  
 @Override  
 public void sendMessage(String message) throws RemoteException {  
 for (ClientProxy clientProxy : clientProxies){  
 clientProxy.receiveMessage(username,message);  
 }  
 }  
 }  
  
 private ArrayList<ClientProxy> clientProxies;  
 private ArrayList<String> usernames;  
  
 private ChatProxy serverChatProxy;  
  
 public ChatServerImpl() throws RemoteException {  
 clientProxies = new ArrayList<>();  
 usernames = new ArrayList<>();  
 serverChatProxy = new ChatProxyImpl("Server");  
 }  
  
 public static void main(String[] args) {  
 try {  
 LocateRegistry.*createRegistry*(Main.*PORT*);  
 ChatServer stub = (ChatServer) new ChatServerImpl();  
 Registry registry = LocateRegistry.*getRegistry*(Main.*PORT*);  
 registry.rebind(Main.*NAME*, stub);  
  
 } catch (RemoteException e) {  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 public ChatProxy subscribeUser(String username, ClientProxy handle) throws RemoteException {  
 System.*out*.printf("%s Subscribed\n",username);  
 clientProxies.add(handle);  
 usernames.add(username);  
 serverChatProxy.sendMessage(String.*format*("%s joined the chat",username));  
 return new ChatProxyImpl(username);  
 }  
  
 @Override  
 public boolean unsubscribeUser(String username) throws RemoteException {  
 for(int i = 0; i < usernames.size(); i++){  
 if(usernames.get(i).equals(username)){  
 usernames.remove(i);  
 clientProxies.remove(i);  
 System.*out*.printf("%s Unsubscribed\n",username);  
 serverChatProxy.sendMessage(String.*format*("%s left the chat",username));  
 return true;  
 }  
 else{  
 System.*err*.printf("%s != %s\n",username,usernames.get(i));  
 }  
 }  
 System.*out*.printf("%s tried to Unsubscribe\n",username);  
 return false;  
 }  
}