CS 3516 Introduction to Computer Networks

Client Server Chat Class Project

Summer 2018

Ying Zhang

Student ID: 8126061018

07/03/2018

It's all my work unless marked to the contrary

Abstract

This report outlines the design and development of a client-server based chatroom project for CS3516. The program was written in Java to run under Mac OS by using Socket to build connection between clients and server. This project allows multiples users connect to a single server at the same time with the localhost port number and their unique nicknames. After they successfully connected, clients can send messages to all the online users in the chatroom. The design and program are modular in nature and make maximum use of abstract data types and of software re-use. A simple GUI was implemented for client by using JFrame under WindowBUilder in Eclipse, and the design was inspired by a Chinese social media provider which named Tencent. Particular attention is paid to the function of multithread and concurrency. The report includes a the test cases used to verify the correct operation of the program, as well as the entire code.

Contents

Project Description

This project allows multiples users connect to a single server at the same time with the specific localhost port number and their unique nicknames. After they successfully connected, clients can send messages to all the online users in the chatroom, and they should be able choose to whisper to a specific person if they wanted to(not working). whisper message can only be seen by the user who the client whispered to. The online user list should be updated and users will be informed every time when someone connect or disconnect(not working).

Detailed Design

The reason why I choose Java to build this project is that in order to have multiple clients connect to a server, multithreads and socket function can be easily used in Java. The overall structure is consist by two parts—the sever part and clients part. A server socket with a specific localhost port number was created, and waiting for client to connect. The creation of SeverSocket require throw Exception, so the try-chat method needs to be implemented in the program. Once a client send the connection request, the serve will use the accept() method and create a thread for the client and wait for the next request. A while loop is implemented in order to continuously read user's message. InputStream() and OutputStream() is used to process the data. The basic structure of my project was base on the socket tutorial video https://www.youtube.com/watch?v=Dy6GfmrqH_I&t=570s.

Summary:

TCP/IP protocol

ServerSocket—>accept() connection

Socket(client)

—> getOutputStream() —write data to a destination

—> getInputStream()—read data from a source

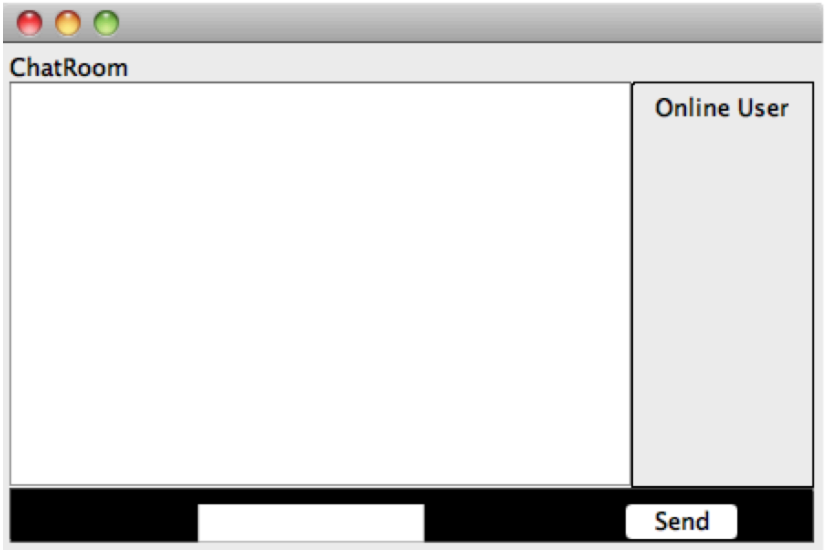
Multiple thread to handle client connection

User to Server—> connection /disconnection, status

User to User —> direct message(whisper), group messages

Server to User —> online/offline

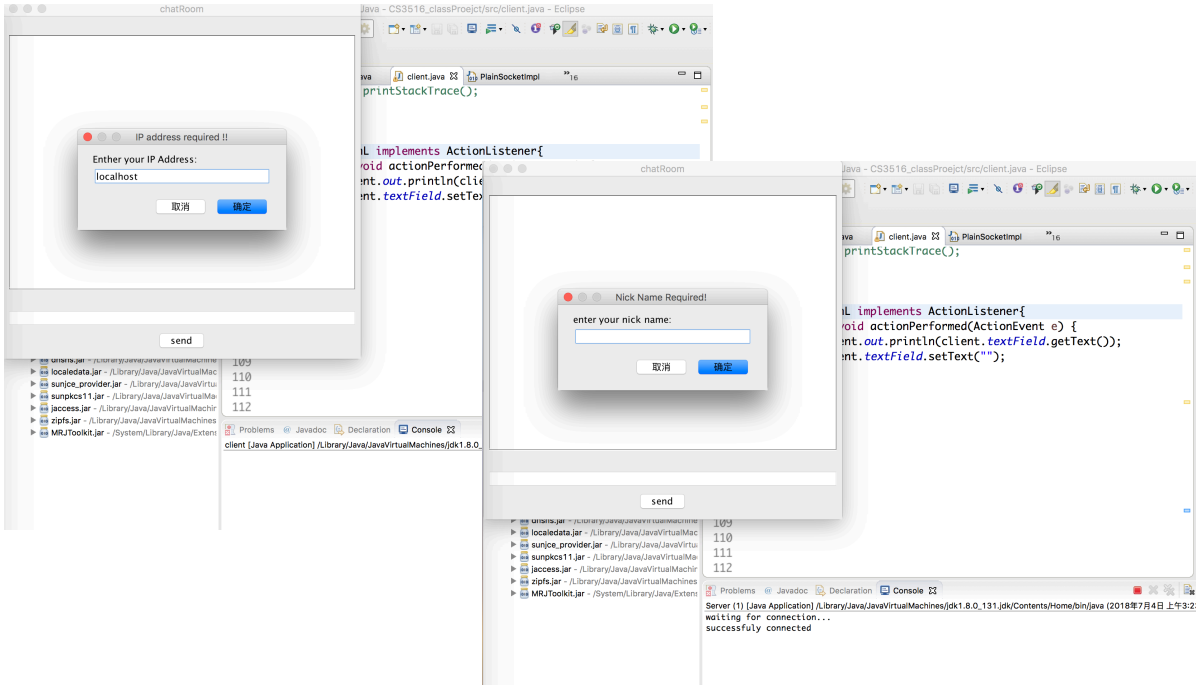
preliminary design of client GUI(generated by WindowBuilder in Eclipse)



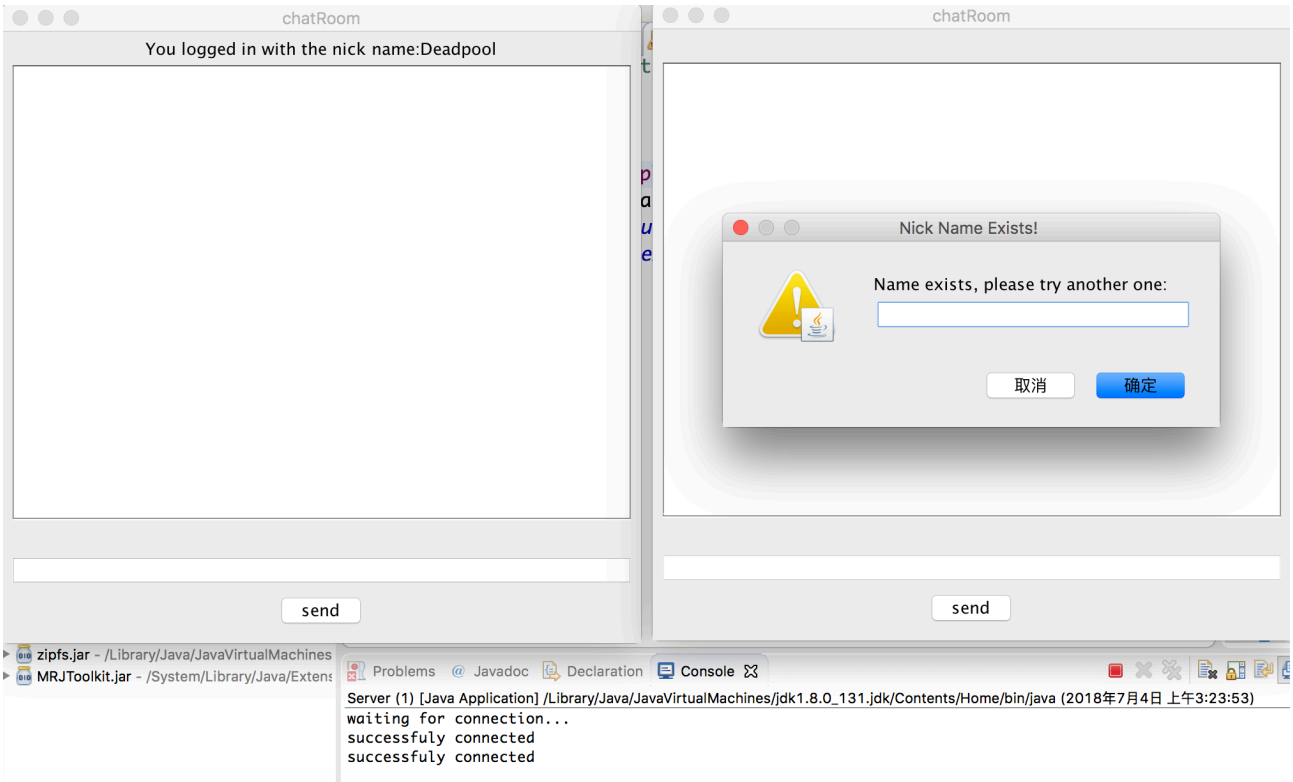
Testing and Evaluation

Client GUI

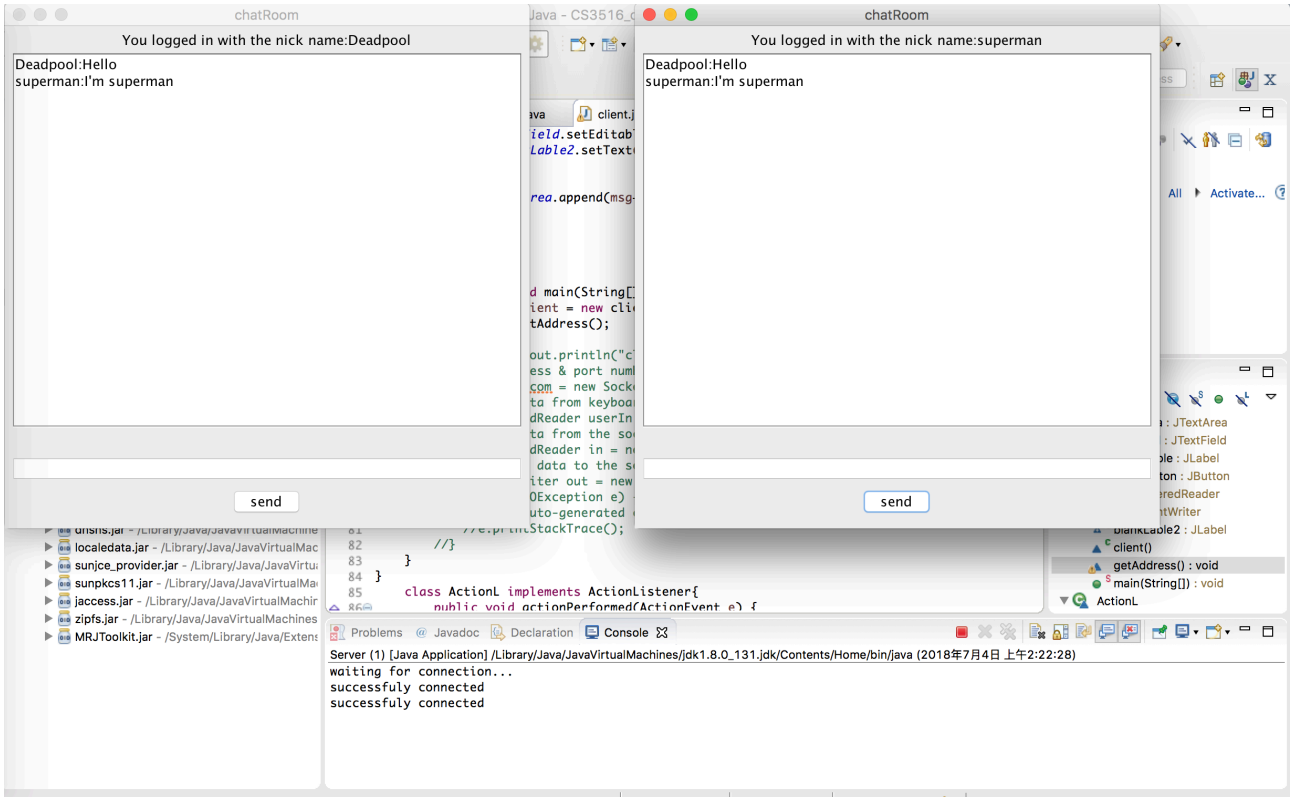
Login with localhost and a unique nick name



Duplicate names:



Sending / Receiving Messages (concurrency)



Future Development

If I had more time working on this I would get the whisper and online user function working with separate interfaces. I would also put all the user information and their message they had sent in the database and allow them to check their chat history whenever they wanted to. Other functions like re-draw the message, or block someone's message can also be implemented.

Appendices

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.ArrayList;

public class Server {
	//unique nick names
	static ArrayList<String> NickNames = new ArrayList<String>();
	//whenever the client send messages to a server, the server need to send
them to all clients
	static ArrayList<PrintWriter> printWriters = new ArrayList<PrintWriter>();
	public static void main(String[] args){
		try {
			System.out.println("waiting for connection...");
			//set port number 3333
			//will be used for accepting incoming client connection
request
			ServerSocket ss = new ServerSocket(3333);
			while(true){
			//Socket for communication
			Socket com = ss.accept();

			//BufferedReader in = new BufferedReader(new
InputStreamReader(com.getInputStream()));
			//send data immediately to the outPutStream
			//PrintWriter out = new
PrintWriter(com.getOutputStream(),true);
			//send data to the stream
			//out.flush();
			System.out.println("successfuly connected");
			comHandler handler = new comHandler(com);
			handler.start();
			}
		} catch (IOException e) {

			e.printStackTrace();
		}
	}
}

class comHandler extends Thread{
	Socket socket;
	BufferedReader in;
	PrintWriter out;
	String nickName;

	public comHandler(Socket socket) throws IOException{
		this.socket = socket;
```

```
	}

	public void run(){
		try{
			in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
			out = new PrintWriter(socket.getOutputStream(),true);

			int count = 0;
			while(true){
				if (count>0) {
					out.println("nick name exists, please try another
one");
				}else{
					out.println("type your nick name");
				}
				nickName= in.readLine();
				if(nickName == null){
					return;
				}
				//break if the nick name is unique
				if(!Server.NickNames.contains(nickName)){
					Server.NickNames.add(nickName);
					break;
				}
				//count > 0 if the name already exist
				count++;

			}
			out.println("Nick Name accepted!"+nickName);
			Server.printWriters.add(out);
			while(true){
				String str = in.readLine();
				if(str == null){
					return;
				}for(PrintWriter pr: Server.printWriters){
					pr.println(nickName + ":" + str);
				}
			}

		}
		catch(Exception e){
			System.out.println(e);
			//add nick name into array list


		}
	}
}

import javax.swing.*;
import javax.swing.border.TitledBorder;

import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.net.UnknownHostException;

public class client {

	static JFrame Window = new JFrame("chatRoom");
	static JTextArea chatArea = new JTextArea(22, 40);
	static JTextField textField = new JTextField(40);
	static JLabel blankLable = new JLabel("           ");
	static JButton sendButton = new JButton("send");
	static BufferedReader in;
	static PrintWriter out;
	static JLabel blankLable2 = new JLabel("         ");
	//static  JScrollPane leftPanel;
	//static JList userList;


	client(){
		Window.setLayout(new FlowLayout());
		Window.add(blankLable2);
		//Window.add(new JScrollPane(leftPanel));
		Window.add(new JScrollPane(chatArea));
		//Window.add(userList);

		Window.add(blankLable);
		Window.add(textField);
		Window.add(sendButton);
		Window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		Window.setSize(500, 500);
		Window.setVisible(true);
		//will not be able to type anything before connection
		textField.setEditable(false);
		chatArea.setEditable(false);
		sendButton.addActionListener(new ActionL());
		textField.addActionListener(new ActionL());
		//leftPanel = new JScrollPane(userList);
		  //leftPanel.setBorder(new TitledBorder("Online user"));

	}
	void getAddress() throws Exception{
		//parentComponent, message, title, messageType
		String IP = JOptionPane.showInputDialog(Window, "Enther your IP
Address:", "IP address required !!",JOptionPane.PLAIN_MESSAGE);
```

```
		Socket com = new Socket(IP,3333);
		in = new BufferedReader(new
InputStreamReader(com.getInputStream()));
		out = new PrintWriter(com.getOutputStream(),true);

		while(true){
			String msg = in.readLine();
			if (msg.equals("type your nick name")){
				String nickName =
JOptionPane.showInputDialog(Window,"enter your nick name: ", "Nick Name
Required! ", JOptionPane.PLAIN_MESSAGE);
				out.println(nickName);
			}else if(msg.equals("nick name exists, please try another
one")){
				String nickName =
JOptionPane.showInputDialog(Window,"Name exists, please try another one: ",
"Nick Name Exists! ", JOptionPane.WARNING_MESSAGE);
				out.println(nickName);
			}else if (msg.startsWith("Nick Name accepted!")){
				textField.setEditable(true);
				blankLable2.setText("You logged in with the nick
name:"+msg.substring(19));

			}else{
				chatArea.append(msg+"\n");
			}

		}
	}

	public static void main(String[] args) throws Exception{
		client chatClient = new client();
		chatClient.getAddress();
		//try {
			//System.out.println("client started");
			//IP address & port number
			//Socket com = new Socket("localhost",3333);
			//read data from keyboard
			//BufferedReader userIn = new BufferedReader(new
InputStreamReader(System.in));
			//read data from the socket input stream
			//BufferedReader in = new BufferedReader(new
InputStreamReader(com.getInputStream()));
		    //sending data to the server/outPutStream
			//PrintWriter out = new
PrintWriter(com.getOutputStream(),true);
		//}  catch (IOException e) {
			// TODO Auto-generated catch block
			//e.printStackTrace();
		//}
	}
}

	class ActionL implements ActionListener{
		public void actionPerformed(ActionEvent e) {
			client.out.println(client.textField.getText());
			client.textField.setText("");
		}
	}
```

Reference:

https://www.youtube.com/watch?v=Dy6GfmrqH_I&t=570s

https://www.javaworld.com/article/2077322/core-java/core-java-sockets-programming-in-java-a-tutorial.html

http://www.baeldung.com/a-guide-to-java-sockets