**NEW BULGARIAN UNIVERSITY**



Department of ***Computer Science***

# COURSE: NETB378 Homework with Programming with JAVA

A Documentation on Assignment:

**Simple chat system**

Andrey Nenchev, F54552

Introduction 3

Server 4

ServerApplication class(the main class) 5

ServerConn class(accepts connections) 6

ClientConn class(listens for client) 7

Client 8

ClientApplication class(the main class) 9

ServerConn class(listens for server) 10

A demonstration of the program 11

Introduction

The task is: Write a chat server and chat client. The server allows multiple clients to connect to the server. Use multi-threading to handle different clients that are connected to the server. Implement GUI for the client part.

There are 4 main problems to this task:

1. Is to establish a port on which the server listens for client connections.
2. To be able to connect client with the server via sockets.
3. To implement somehow on both sides sending and receiving to be available at the same time.
4. To implement communication between clients.

The task is completed with two projects:

1. ServerChatApplication Project
2. ClientChatApplication Project

Server

The Server works with 2 main threads:

1. When we run the application the main thread runs.
2. A new thread is established for listening for clients connections. So the standard input is free for the main thread.

**For each connection to the Server a new thread is created**. Every client runs on separate thread.

For this purpose 3 classes are implemented:

Class ServerApplication – The main class. Responsible for the visual form of application and process input.

Class ServerConn – The server class. Responsible for listening and accepts connections.

Class ClientConn – The client class. Responsible for listening for the client and outputting it in the form.

ServerApplication class

The client – server chat works on port “4444”.

For this purpose:

1. int port = 4444’.
2. Hashtable<Thread, Socket> map: Keeps all client’s threads and sockets.
3. Hashtable<String, PrintWriter> clientsOutput: Client’s thread names and PrintWriters

First The server opens Server Socket on port 4444. If it’s successful starts a new thread with class ServerConn and attributes the Server socket.

Thread t = new Thread(new ServerConn(server));

t.start();

If there are clients connected to the server, they are listed in the listclients ListBox section.

There is an event listener for text field jtextinput, listens for key release:

private void jTextInputKeyReleased(java.awt.event.KeyEvent evt)

if the key is “Enter” a message is send to all selected clients.

ServerConn class

ServerConn class is responsible only for accepting the connections.

This is done by an infinite loop :

while(true)

client = this.server.accept();

In this loop for each client connection a new thread with class ClientConn and attributes Socket is created :

Thread t = new Thread(new ClientConn(client));

and added to ServerChet.map HashMap.

serverchat.ServerChat.addThread(t, client);

ClientConn class (package serverchat)

ClientConn class uses the client Socket from the constructor to create an input streamer and put it in a buffer reader:

private BufferedReader in = new BufferedReader(new InputStreamReader( client.getInputStream()));

The thread reads from the buffer reader in a while loop:

while (condition)) {

ServerApplication.ServApp.printMsgOnScreen( “message”);

}

The conditions are:

1. If the message is “TERMINATE” : Application stops
2. <<REFRESH>> : Server automatically refreshes the client’s “list of clients”
3. <<SENDTO>> : Server automatically resends the message to the other party.

Client

The Client works with only 2 threads:

1. When the application is running the main thread runs establishing connection with the server.
2. A new thread is created for listening to the Server. So the standard input is free for the main thread.

For this purpose 2 classes are implemented:

Class ClientChat – The main class. Responsible for establishing connection with the server and the Input.

Class ServerConn – The server class. Responsible for listening to the server and outputting it in jtextoutput text field.

ClientApplication class

The client – server chat works on port “4444”.

For this purpose:

1. int port = 4444
2. String host = “localhost”

First The client opens a Socket on port 4444. If it’s successful starts a new thread with class ServerConn and attributes the socket.

Thread t = new Thread(new ServerConn(server));

t.start();

There is an event listener attached to the jtextinput text field:

*private void jTextInputKeyReleased(java.awt.event.KeyEvent evt)*

if the key “Enter” is released the message is send to the server.

If clients are selected a “<<SENDTO>>” is attached to the message.

ServerConn class

ServerConn class uses the server Socket from the constructor to create an input streamer and put it in a buffer reader:

private BufferedReader in = new BufferedReader(new InputStreamReader( server.getInputStream()));

The thread reads from the buffer reader in a while loop:

while (msg = in.readLine())) {

ClientApplication.clientApp.printMsgOnScreen("message”);

}

There are conditions:

1. <<AddClient>> : A client is added to listclients ListBox
2. <<RemoveClient>> : A client is removed from the listclients ListBox
3. <<FROMCLIENT>> : the message is printed on the screen from the other party
4. Else the message is from server and is printed on the screen with “Server” attached in front of it.

A demonstration of the program:

