**Chat Server and Chat Client In Java**

A simple chat server that can serve multiple clients using Socket programming and multithreading in Java. Each client can send message to server, which then broadcasts the message to all other clients currently connected.

Each client is identified by a name, which is taken from user at the time of starting the client.

**Chat Server**

The following is the code for **ChatServer.java.** Compile and run it first.

// Chat Server runs at port no. 9999

import java.io.\*;

import java.util.\*;

import java.net.\*;

import static java.lang.System.out;

public class ChatServer {

Vector<String> users = new Vector<String>();

Vector<HandleClient> clients = new Vector<HandleClient>();

public void process() throws Exception {

ServerSocket server = new ServerSocket(9999,10);

out.println("Server Started...");

while( true) {

Socket client = server.accept();

HandleClient c = new HandleClient(client);

clients.add(c);

} // end of while

}

public static void main(String ... args) throws Exception {

new ChatServer().process();

} // end of main

public void boradcast(String user, String message) {

// send message to all connected users

for ( HandleClient c : clients )

if ( ! c.getUserName().equals(user) )

c.sendMessage(user,message);

}

class HandleClient extends Thread {

String name = "";

BufferedReader input;

PrintWriter output;

public HandleClient(Socket client) throws Exception {

// get input and output streams

input = new BufferedReader( new InputStreamReader( client.getInputStream())) ;

output = new PrintWriter ( client.getOutputStream(),true);

// read name

name = input.readLine();

users.add(name); // add to vector

start();

}

public void sendMessage(String uname,String msg) {

output.println( uname + ":" + msg);

}

public String getUserName() {

return name;

}

public void run() {

String line;

try {

while(true) {

line = input.readLine();

if ( line.equals("end") ) {

clients.remove(this);

users.remove(name);

break;

}

boradcast(name,line); // method of outer class - send messages to all

} // end of while

} // try

catch(Exception ex) {

System.out.println(ex.getMessage());

}

} // end of run()

} // end of inner class

} // end of Server

**Chat Client**

The following is the code for **ChatClient.java**. It prompt user to enter name and identifies the client with that name. Server is assumed to run on the same system (localhost) at port number 9999. Run client once for each user by going to a different command prompt.

import java.io.\*;

import java.util.\*;

import java.net.\*;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

import static java.lang.System.out;

public class ChatClient extends JFrame implements ActionListener {

String uname;

PrintWriter pw;

BufferedReader br;

JTextArea taMessages;

JTextField tfInput;

JButton btnSend,btnExit;

Socket client;

public ChatClient(String uname,String servername) throws Exception {

super(uname); // set title for frame

this.uname = uname;

client = new Socket(servername,9999);

br = new BufferedReader( new InputStreamReader( client.getInputStream()) ) ;

pw = new PrintWriter(client.getOutputStream(),true);

pw.println(uname); // send name to server

buildInterface();

new MessagesThread().start(); // create thread for listening for messages

}

public void buildInterface() {

btnSend = new JButton("Send");

btnExit = new JButton("Exit");

taMessages = new JTextArea();

taMessages.setRows(10);

taMessages.setColumns(50);

taMessages.setEditable(false);

tfInput = new JTextField(50);

JScrollPane sp = new JScrollPane(taMessages, JScrollPane.VERTICAL\_SCROLLBAR\_AS\_NEEDED,

JScrollPane.HORIZONTAL\_SCROLLBAR\_AS\_NEEDED);

add(sp,"Center");

JPanel bp = new JPanel( new FlowLayout());

bp.add(tfInput);

bp.add(btnSend);

bp.add(btnExit);

add(bp,"South");

btnSend.addActionListener(this);

btnExit.addActionListener(this);

setSize(500,300);

setVisible(true);

pack();

}

public void actionPerformed(ActionEvent evt) {

if ( evt.getSource() == btnExit ) {

pw.println("end"); // send end to server so that server know about the termination

System.exit(0);

} else {

// send message to server

pw.println(tfInput.getText());

}

}

public static void main(String ... args) {

// take username from user

String name = JOptionPane.showInputDialog(null,"Enter your name :", "Username",

JOptionPane.PLAIN\_MESSAGE);

String servername = "localhost";

try {

new ChatClient( name ,servername);

} catch(Exception ex) {

out.println( "Error --> " + ex.getMessage());

}

} // end of main

// inner class for Messages Thread

class MessagesThread extends Thread {

public void run() {

String line;

try {

while(true) {

line = br.readLine();

taMessages.append(line + "\n");

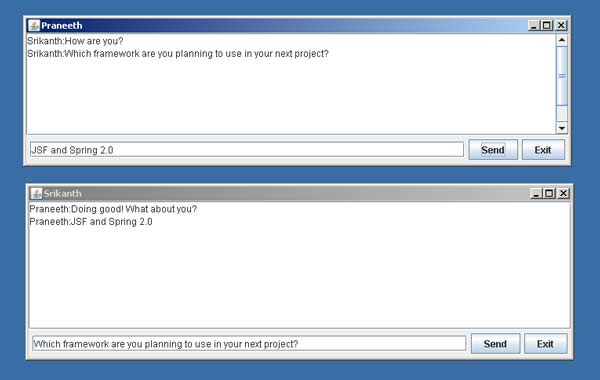
} // end of while

} catch(Exception ex) {}

}

}

} // end of client

The following screenshot shows the program in execution with two clients.  
  
  
  
**Srikanth.**