**A chat application with multi-users and file sharing**

It comprises of two stand-alone applications:

* The Server
* The Client / Messenger

**The Server**

This is where the service is started and waits for the clients to connect to. Here the login details of all users are controlled; it opens a specified port on the computer to allow the other clients to connect through in order to exchange data. It makes use of a xml file to start and save those login details.

The server has to be running on one of the networked computers whose internet proxy address must be known for other clients to connect to.

**Sources**

The Source code for the server application is made up of 4 classes

1. Database.java
2. Message.java
3. DatabaseFrame.java
4. SocketServer.java

Also the Database file in xml format

1. Data.xml

Database.java: This class is responsible for managing the users that are allowed to connect to the chat application. Here is the source

package com.socket;

import java.io.\*;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.transform.Transformer;

import javax.xml.transform.TransformerFactory;

import javax.xml.transform.dom.DOMSource;

import javax.xml.transform.stream.StreamResult;

import org.w3c.dom.\*;

public class Database {

public String filePath;

public Database(String filePath){

this.filePath = filePath;

}

public boolean userExists(String username){

try{

File fXmlFile = new File(filePath);

DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

Document doc = dBuilder.parse(fXmlFile);

doc.getDocumentElement().normalize();

NodeList nList = doc.getElementsByTagName("user");

for (int temp = 0; temp < nList.getLength(); temp++) {

Node nNode = nList.item(temp);

if (nNode.getNodeType() == Node.ELEMENT\_NODE) {

Element eElement = (Element) nNode;

if(getTagValue("username", eElement).equals(username)){

return true;

}

}

}

return false;

}

catch(Exception ex){

System.out.println("Database exception : userExists()");

return false;

}

}

public boolean checkLogin(String username, String password){

if(!userExists(username)){ return false; }

try{

File fXmlFile = new File(filePath);

DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

Document doc = dBuilder.parse(fXmlFile);

doc.getDocumentElement().normalize();

NodeList nList = doc.getElementsByTagName("user");

for (int temp = 0; temp < nList.getLength(); temp++) {

Node nNode = nList.item(temp);

if (nNode.getNodeType() == Node.ELEMENT\_NODE) {

Element eElement = (Element) nNode;

if(getTagValue("username", eElement).equals(username) && getTagValue("password", eElement).equals(password)){

return true;

}

}

}

System.out.println("Hippie");

return false;

}

catch(Exception ex){

System.out.println("Database exception : userExists()");

return false;

}

}

public void addUser(String username, String password){

try {

DocumentBuilderFactory docFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder docBuilder = docFactory.newDocumentBuilder();

Document doc = docBuilder.parse(filePath);

Node data = doc.getFirstChild();

Element newuser = doc.createElement("user");

Element newusername = doc.createElement("username"); newusername.setTextContent(username);

Element newpassword = doc.createElement("password"); newpassword.setTextContent(password);

newuser.appendChild(newusername); newuser.appendChild(newpassword); data.appendChild(newuser);

TransformerFactory transformerFactory = TransformerFactory.newInstance();

Transformer transformer = transformerFactory.newTransformer();

DOMSource source = new DOMSource(doc);

StreamResult result = new StreamResult(new File(filePath));

transformer.transform(source, result);

}

catch(Exception ex){

System.out.println("Exceptionmodify xml");

}

}

public static String getTagValue(String sTag, Element eElement) {

NodeList nlList = eElement.getElementsByTagName(sTag).item(0).getChildNodes();

Node nValue = (Node) nlList.item(0);

return nValue.getNodeValue();

}

}

Message.java: This class handles messages that are coming from various clients, both the general and private and makes them available to the respective client.

The source goes thus:

package com.socket;

import java.io.Serializable;

public class Message implements Serializable{

private static final long serialVersionUID = 1L;

public String type, sender, content, recipient;

public Message(String type, String sender, String content, String recipient){

this.type = type; this.sender = sender; this.content = content; this.recipient = recipient;

}

@Override

public String toString(){

return "{type='"+type+"', sender='"+sender+"', content='"+content+"', recipient='"+recipient+"'}";

}

}

DatabaseFrame.java: It would be nice if the progress of the activities running in the background is being displayed to the administrator so as to easily track down any error. This class does the displaying and here is the source:

package com.socket;

import java.awt.Color;

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

import java.io.File;

import javax.swing.JFileChooser;

import javax.swing.UIManager;

public class ServerFrame extends javax.swing.JFrame {

public SocketServer server;

public Thread serverThread;

public String filePath = "D:/Data.xml";

public JFileChooser fileChooser;

public ServerFrame() {

initComponents();

jTextField3.setEditable(false);

jTextField3.setBackground(Color.WHITE);

fileChooser = new JFileChooser();

jTextArea1.setEditable(false);

}

public boolean isWin32(){

return System.getProperty("os.name").startsWith("Windows");

}

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents

private void initComponents() {

jButton1 = new javax.swing.JButton();

jScrollPane1 = new javax.swing.JScrollPane();

jTextArea1 = new javax.swing.JTextArea();

jLabel3 = new javax.swing.JLabel();

jTextField3 = new javax.swing.JTextField();

jButton2 = new javax.swing.JButton();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

setTitle("jServer");

jButton1.setText("Start Server");

jButton1.setEnabled(false);

jButton1.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton1ActionPerformed(evt);

}

});

jTextArea1.setColumns(20);

jTextArea1.setFont(new java.awt.Font("Consolas", 0, 12)); // NOI18N

jTextArea1.setRows(5);

jScrollPane1.setViewportView(jTextArea1);

jLabel3.setText("Database File : ");

jButton2.setText("Browse...");

jButton2.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton2ActionPerformed(evt);

}

});

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jScrollPane1)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()

.addComponent(jLabel3)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addComponent(jTextField3, javax.swing.GroupLayout.DEFAULT\_SIZE, 282, Short.MAX\_VALUE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(jButton2, javax.swing.GroupLayout.PREFERRED\_SIZE, 91, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(jButton1)))

.addContainerGap())

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jTextField3, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabel3)

.addComponent(jButton2)

.addComponent(jButton1))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT\_SIZE, 287, Short.MAX\_VALUE)

.addContainerGap())

);

pack();

}// </editor-fold>//GEN-END:initComponents

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_jButton1ActionPerformed

server = new SocketServer(this);

jButton1.setEnabled(false); jButton2.setEnabled(false);

}//GEN-LAST:event\_jButton1ActionPerformed

public void RetryStart(int port){

if(server != null){ server.stop(); }

server = new SocketServer(this, port);

}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_jButton2ActionPerformed

fileChooser.showDialog(this, "Select");

File file = fileChooser.getSelectedFile();

if(file != null){

filePath = file.getPath();

if(this.isWin32()){ filePath = filePath.replace("\\", "/"); }

jTextField3.setText(filePath);

jButton1.setEnabled(true);

}

}//GEN-LAST:event\_jButton2ActionPerformed

public static void main(String args[]) {

try{

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());

}

catch(Exception ex){

System.out.println("Look & Feel Exception");

}

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

new ServerFrame().setVisible(true);

}

});

}

// Variables declaration - do not modify//GEN-BEGIN:variables

private javax.swing.JButton jButton1;

private javax.swing.JButton jButton2;

private javax.swing.JLabel jLabel3;

private javax.swing.JScrollPane jScrollPane1;

public javax.swing.JTextArea jTextArea1;

private javax.swing.JTextField jTextField3;

// End of variables declaration//GEN-END:variables

}

SocketServer.java: Opening of Threads for various clients, opening port for other computers to connect, connecting one client to another, finding number of connected users and pushing messages to appropriate user. These are all what does down here in the SocketServer class with this source:

package com.socket;

import java.io.\*;

import java.net.\*;

class ServerThread extends Thread {

public SocketServer server = null;

public Socket socket = null;

public int ID = -1;

public String username = "";

public ObjectInputStream streamIn = null;

public ObjectOutputStream streamOut = null;

public ServerFrame ui;

public ServerThread(SocketServer \_server, Socket \_socket){

super();

server = \_server;

socket = \_socket;

ID = socket.getPort();

ui = \_server.ui;

}

public void send(Message msg){

try {

streamOut.writeObject(msg);

streamOut.flush();

}

catch (IOException ex) {

System.out.println("Exception [SocketClient : send(...)]");

}

}

public int getID(){

return ID;

}

@SuppressWarnings("deprecation")

public void run(){

ui.jTextArea1.append("\nServer Thread " + ID + " running.");

while (true){

try{

Message msg = (Message) streamIn.readObject();

server.handle(ID, msg);

}

catch(Exception ioe){

System.out.println(ID + " ERROR reading: " + ioe.getMessage());

server.remove(ID);

stop();

}

}

}

public void open() throws IOException {

streamOut = new ObjectOutputStream(socket.getOutputStream());

streamOut.flush();

streamIn = new ObjectInputStream(socket.getInputStream());

}

public void close() throws IOException {

if (socket != null) socket.close();

if (streamIn != null) streamIn.close();

if (streamOut != null) streamOut.close();

}

}

public class SocketServer implements Runnable {

public ServerThread clients[];

public ServerSocket server = null;

public Thread thread = null;

public int clientCount = 0, port = 13000;

public ServerFrame ui;

public Database db;

public SocketServer(ServerFrame frame){

clients = new ServerThread[50];

ui = frame;

db = new Database(ui.filePath);

try{

server = new ServerSocket(port);

port = server.getLocalPort();

ui.jTextArea1.append("Server startet. IP : " + InetAddress.getLocalHost() + ", Port : " + server.getLocalPort());

start();

}

catch(IOException ioe){

ui.jTextArea1.append("Can not bind to port : " + port + "\nRetrying");

ui.RetryStart(0);

}

}

public SocketServer(ServerFrame frame, int Port){

clients = new ServerThread[50];

ui = frame;

port = Port;

db = new Database(ui.filePath);

try{

server = new ServerSocket(port);

port = server.getLocalPort();

ui.jTextArea1.append("Server startet. IP : " + InetAddress.getLocalHost() + ", Port : " + server.getLocalPort());

start();

}

catch(IOException ioe){

ui.jTextArea1.append("\nCan not bind to port " + port + ": " + ioe.getMessage());

}

}

public void run(){

while (thread != null){

try{

ui.jTextArea1.append("\nWaiting for a client ...");

addThread(server.accept());

}

catch(Exception ioe){

ui.jTextArea1.append("\nServer accept error: \n");

ui.RetryStart(0);

}

}

}

public void start(){

if (thread == null){

thread = new Thread(this);

thread.start();

}

}

@SuppressWarnings("deprecation")

public void stop(){

if (thread != null){

thread.stop();

thread = null;

}

}

private int findClient(int ID){

for (int i = 0; i < clientCount; i++){

if (clients[i].getID() == ID){

return i;

}

}

return -1;

}

public synchronized void handle(int ID, Message msg){

if (msg.content.equals(".bye")){

Announce("signout", "SERVER", msg.sender);

remove(ID);

}

else{

if(msg.type.equals("login")){

if(findUserThread(msg.sender) == null){

if(db.checkLogin(msg.sender, msg.content)){

clients[findClient(ID)].username = msg.sender;

clients[findClient(ID)].send(new Message("login", "SERVER", "TRUE", msg.sender));

Announce("newuser", "SERVER", msg.sender);

SendUserList(msg.sender);

}

else{

clients[findClient(ID)].send(new Message("login", "SERVER", "FALSE", msg.sender));

}

}

else{

clients[findClient(ID)].send(new Message("login", "SERVER", "FALSE", msg.sender));

}

}

else if(msg.type.equals("message")){

if(msg.recipient.equals("All")){

Announce("message", msg.sender, msg.content);

}

else{

findUserThread(msg.recipient).send(new Message(msg.type, msg.sender, msg.content, msg.recipient));

clients[findClient(ID)].send(new Message(msg.type, msg.sender, msg.content, msg.recipient));

}

}

else if(msg.type.equals("test")){

clients[findClient(ID)].send(new Message("test", "SERVER", "OK", msg.sender));

}

else if(msg.type.equals("signup")){

if(findUserThread(msg.sender) == null){

if(!db.userExists(msg.sender)){

db.addUser(msg.sender, msg.content);

clients[findClient(ID)].username = msg.sender;

clients[findClient(ID)].send(new Message("signup", "SERVER", "TRUE", msg.sender));

clients[findClient(ID)].send(new Message("login", "SERVER", "TRUE", msg.sender));

Announce("newuser", "SERVER", msg.sender);

SendUserList(msg.sender);

}

else{

clients[findClient(ID)].send(new Message("signup", "SERVER", "FALSE", msg.sender));

}

}

else{

clients[findClient(ID)].send(new Message("signup", "SERVER", "FALSE", msg.sender));

}

}

else if(msg.type.equals("upload\_req")){

if(msg.recipient.equals("All")){

clients[findClient(ID)].send(new Message("message", "SERVER", "Uploading to 'All' forbidden", msg.sender));

}

else{

findUserThread(msg.recipient).send(new Message("upload\_req", msg.sender, msg.content, msg.recipient));

}

}

else if(msg.type.equals("upload\_res")){

if(!msg.content.equals("NO")){

String IP = findUserThread(msg.sender).socket.getInetAddress().getHostAddress();

findUserThread(msg.recipient).send(new Message("upload\_res", IP, msg.content, msg.recipient));

}

else{

findUserThread(msg.recipient).send(new Message("upload\_res", msg.sender, msg.content, msg.recipient));

}

}

}

}

public void Announce(String type, String sender, String content){

Message msg = new Message(type, sender, content, "All");

for(int i = 0; i < clientCount; i++){

clients[i].send(msg);

}

}

public void SendUserList(String toWhom){

for(int i = 0; i < clientCount; i++){

findUserThread(toWhom).send(new Message("newuser", "SERVER", clients[i].username, toWhom));

}

}

public ServerThread findUserThread(String usr){

for(int i = 0; i < clientCount; i++){

if(clients[i].username.equals(usr)){

return clients[i];

}

}

return null;

}

@SuppressWarnings("deprecation")

public synchronized void remove(int ID){

int pos = findClient(ID);

if (pos >= 0){

ServerThread toTerminate = clients[pos];

ui.jTextArea1.append("\nRemoving client thread " + ID + " at " + pos);

if (pos < clientCount-1){

for (int i = pos+1; i < clientCount; i++){

clients[i-1] = clients[i];

}

}

clientCount--;

try{

toTerminate.close();

}

catch(IOException ioe){

ui.jTextArea1.append("\nError closing thread: " + ioe);

}

toTerminate.stop();

}

}

private void addThread(Socket socket){

if (clientCount < clients.length){

ui.jTextArea1.append("\nClient accepted: " + socket);

clients[clientCount] = new ServerThread(this, socket);

try{

clients[clientCount].open();

clients[clientCount].start();

clientCount++;

}

catch(IOException ioe){

ui.jTextArea1.append("\nError opening thread: " + ioe);

}

}

else{

ui.jTextArea1.append("\nClient refused: maximum " + clients.length + " reached.");

}

}

}

Data.xml: A simple that is to be imported will starting the server and subsequent registered users are automatically save in it for further use. The source goes thus:

<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<data>

<user><username>Daniel</username><password>1234</password></user>

</data>

**The Client / Messenger**

This is the second application that is run on the client’s machine for connecting, chatting viewing people online, viewing public and private message, sending and receiving conversations, sending and receiving file and documents. The clients has to know the internet proxy address of the server and the port it allows connection on for the purpose of the chat before it allow any user to register or login.

**Sources:**

The classes are grouped into 2 packages based on their functions. Those doing the display and those doing the background works.

1. com.socket
2. com.ui

* **com.socket**

In the com.socket package we have the following classes there in:

1. Download.java
2. History.java
3. Message.java
4. SocketClient.java
5. Upload.java

**Download.java**: This class is handles receiving of file from another client.

package com.socket;

import com.ui.ChatFrame;

import java.io.\*;

import java.net.\*;

import java.util.logging.Level;

import java.util.logging.Logger;

public class Download implements Runnable{

public ServerSocket server;

public Socket socket;

public int port;

public String saveTo = "";

public InputStream In;

public FileOutputStream Out;

public ChatFrame ui;

public Download(String saveTo, ChatFrame ui){

try {

server = new ServerSocket(0);

port = server.getLocalPort();

this.saveTo = saveTo;

this.ui = ui;

}

catch (IOException ex) {

System.out.println("Exception [Download : Download(...)]");

}

}

@Override

public void run() {

try {

socket = server.accept();

System.out.println("Download : "+socket.getRemoteSocketAddress());

In = socket.getInputStream();

Out = new FileOutputStream(saveTo);

byte[] buffer = new byte[1024];

int count;

while((count = In.read(buffer)) >= 0){

Out.write(buffer, 0, count);

}

Out.flush();

ui.jTextArea1.append("[Application > Me] : Download complete\n");

if(Out != null){ Out.close(); }

if(In != null){ In.close(); }

if(socket != null){ socket.close(); }

}

catch (Exception ex) {

System.out.println("Exception [Download : run(...)]");

}

}

}

**History.java**: All conversations are kept with this class for a record purpose.

package com.socket;

import java.io.\*;

import java.util.ArrayList;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.transform.Transformer;

import javax.xml.transform.TransformerFactory;

import javax.xml.transform.dom.DOMSource;

import javax.xml.transform.stream.StreamResult;

import org.w3c.dom.\*;

import com.ui.HistoryFrame;

import javax.swing.table.DefaultTableModel;

public class History {

public String filePath;

public History(String filePath){

this.filePath = filePath;

}

public void addMessage(Message msg, String time){

try {

DocumentBuilderFactory docFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder docBuilder = docFactory.newDocumentBuilder();

Document doc = docBuilder.parse(filePath);

Node data = doc.getFirstChild();

Element message = doc.createElement("message");

Element \_sender = doc.createElement("sender"); \_sender.setTextContent(msg.sender);

Element \_content = doc.createElement("content"); \_content.setTextContent(msg.content);

Element \_recipient = doc.createElement("recipient"); \_recipient.setTextContent(msg.recipient);

Element \_time = doc.createElement("time"); \_time.setTextContent(time);

message.appendChild(\_sender); message.appendChild(\_content); message.appendChild(\_recipient); message.appendChild(\_time);

data.appendChild(message);

TransformerFactory transformerFactory = TransformerFactory.newInstance();

Transformer transformer = transformerFactory.newTransformer();

DOMSource source = new DOMSource(doc);

StreamResult result = new StreamResult(new File(filePath));

transformer.transform(source, result);

}

catch(Exception ex){

System.out.println("Exceptionmodify xml");

}

}

public void FillTable(HistoryFrame frame){

DefaultTableModel model = (DefaultTableModel) frame.jTable1.getModel();

try{

File fXmlFile = new File(filePath);

DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

Document doc = dBuilder.parse(fXmlFile);

doc.getDocumentElement().normalize();

NodeList nList = doc.getElementsByTagName("message");

for (int temp = 0; temp < nList.getLength(); temp++) {

Node nNode = nList.item(temp);

if (nNode.getNodeType() == Node.ELEMENT\_NODE) {

Element eElement = (Element) nNode;

model.addRow(new Object[]{getTagValue("sender", eElement), getTagValue("content", eElement), getTagValue("recipient", eElement), getTagValue("time", eElement)});

}

}

}

catch(Exception ex){

System.out.println("Filling Exception");

}

}

public static String getTagValue(String sTag, Element eElement) {

NodeList nlList = eElement.getElementsByTagName(sTag).item(0).getChildNodes();

Node nValue = (Node) nlList.item(0);

return nValue.getNodeValue();

}

}

**Message.java**: This handles sending of chat conversations from one client to another.

package com.socket;

import java.io.\*;

import java.util.ArrayList;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.transform.Transformer;

import javax.xml.transform.TransformerFactory;

import javax.xml.transform.dom.DOMSource;

import javax.xml.transform.stream.StreamResult;

import org.w3c.dom.\*;

import com.ui.HistoryFrame;

import javax.swing.table.DefaultTableModel;

public class History {

public String filePath;

public History(String filePath){

this.filePath = filePath;

}

public void addMessage(Message msg, String time){

try {

DocumentBuilderFactory docFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder docBuilder = docFactory.newDocumentBuilder();

Document doc = docBuilder.parse(filePath);

Node data = doc.getFirstChild();

Element message = doc.createElement("message");

Element \_sender = doc.createElement("sender"); \_sender.setTextContent(msg.sender);

Element \_content = doc.createElement("content"); \_content.setTextContent(msg.content);

Element \_recipient = doc.createElement("recipient"); \_recipient.setTextContent(msg.recipient);

Element \_time = doc.createElement("time"); \_time.setTextContent(time);

message.appendChild(\_sender); message.appendChild(\_content); message.appendChild(\_recipient); message.appendChild(\_time);

data.appendChild(message);

TransformerFactory transformerFactory = TransformerFactory.newInstance();

Transformer transformer = transformerFactory.newTransformer();

DOMSource source = new DOMSource(doc);

StreamResult result = new StreamResult(new File(filePath));

transformer.transform(source, result);

}

catch(Exception ex){

System.out.println("Exceptionmodify xml");

}

}

public void FillTable(HistoryFrame frame){

DefaultTableModel model = (DefaultTableModel) frame.jTable1.getModel();

try{

File fXmlFile = new File(filePath);

DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

Document doc = dBuilder.parse(fXmlFile);

doc.getDocumentElement().normalize();

NodeList nList = doc.getElementsByTagName("message");

for (int temp = 0; temp < nList.getLength(); temp++) {

Node nNode = nList.item(temp);

if (nNode.getNodeType() == Node.ELEMENT\_NODE) {

Element eElement = (Element) nNode;

model.addRow(new Object[]{getTagValue("sender", eElement), getTagValue("content", eElement), getTagValue("recipient", eElement), getTagValue("time", eElement)});

}

}

}

catch(Exception ex){

System.out.println("Filling Exception");

}

}

public static String getTagValue(String sTag, Element eElement) {

NodeList nlList = eElement.getElementsByTagName(sTag).item(0).getChildNodes();

Node nValue = (Node) nlList.item(0);

return nValue.getNodeValue();

}

}

**SocketClient.java**: This class is responsible for establishing any connection including for messages, file transfer, and any other.

package com.socket;

import com.ui.ChatFrame;

import java.io.\*;

import java.net.\*;

import java.util.Date;

import javax.swing.JFileChooser;

import javax.swing.JOptionPane;

import javax.swing.table.DefaultTableModel;

public class SocketClient implements Runnable{

public int port;

public String serverAddr;

public Socket socket;

public ChatFrame ui;

public ObjectInputStream In;

public ObjectOutputStream Out;

public History hist;

public SocketClient(ChatFrame frame) throws IOException{

ui = frame; this.serverAddr = ui.serverAddr; this.port = ui.port;

socket = new Socket(InetAddress.getByName(serverAddr), port);

Out = new ObjectOutputStream(socket.getOutputStream());

Out.flush();

In = new ObjectInputStream(socket.getInputStream());

hist = ui.hist;

}

@Override

public void run() {

boolean keepRunning = true;

while(keepRunning){

try {

Message msg = (Message) In.readObject();

System.out.println("Incoming : "+msg.toString());

if(msg.type.equals("message")){

if(msg.recipient.equals(ui.username)){

ui.jTextArea1.append("["+msg.sender +" > Me] : " + msg.content + "\n");

}

else{

ui.jTextArea1.append("["+ msg.sender +" > "+ msg.recipient +"] : " + msg.content + "\n");

}

if(!msg.content.equals(".bye") && !msg.sender.equals(ui.username)){

String msgTime = (new Date()).toString();

try{

hist.addMessage(msg, msgTime);

DefaultTableModel table = (DefaultTableModel) ui.historyFrame.jTable1.getModel();

table.addRow(new Object[]{msg.sender, msg.content, "Me", msgTime});

}

catch(Exception ex){}

}

}

else if(msg.type.equals("login")){

if(msg.content.equals("TRUE")){

ui.jButton2.setEnabled(false); ui.jButton3.setEnabled(false);

ui.jButton4.setEnabled(true); ui.jButton5.setEnabled(true);

ui.jTextArea1.append("[SERVER > Me] : Login Successful\n");

ui.jTextField3.setEnabled(false); ui.jPasswordField1.setEnabled(false);

}

else{

ui.jTextArea1.append("[SERVER > Me] : Login Failed\n");

}

}

else if(msg.type.equals("test")){

ui.jButton1.setEnabled(false);

ui.jButton2.setEnabled(true); ui.jButton3.setEnabled(true);

ui.jTextField3.setEnabled(true); ui.jPasswordField1.setEnabled(true);

ui.jTextField1.setEditable(false); ui.jTextField2.setEditable(false);

ui.jButton7.setEnabled(true);

}

else if(msg.type.equals("newuser")){

if(!msg.content.equals(ui.username)){

boolean exists = false;

for(int i = 0; i < ui.model.getSize(); i++){

if(ui.model.getElementAt(i).equals(msg.content)){

exists = true; break;

}

}

if(!exists){ ui.model.addElement(msg.content); }

}

}

else if(msg.type.equals("signup")){

if(msg.content.equals("TRUE")){

ui.jButton2.setEnabled(false); ui.jButton3.setEnabled(false);

ui.jButton4.setEnabled(true); ui.jButton5.setEnabled(true);

ui.jTextArea1.append("[SERVER > Me] : Singup Successful\n");

}

else{

ui.jTextArea1.append("[SERVER > Me] : Signup Failed\n");

}

}

else if(msg.type.equals("signout")){

if(msg.content.equals(ui.username)){

ui.jTextArea1.append("["+ msg.sender +" > Me] : Bye\n");

ui.jButton1.setEnabled(true); ui.jButton4.setEnabled(false);

ui.jTextField1.setEditable(true); ui.jTextField2.setEditable(true);

for(int i = 1; i < ui.model.size(); i++){

ui.model.removeElementAt(i);

}

ui.clientThread.stop();

}

else{

ui.model.removeElement(msg.content);

ui.jTextArea1.append("["+ msg.sender +" > All] : "+ msg.content +" has signed out\n");

}

}

else if(msg.type.equals("upload\_req")){

if(JOptionPane.showConfirmDialog(ui, ("Accept '"+msg.content+"' from "+msg.sender+" ?")) == 0){

JFileChooser jf = new JFileChooser();

jf.setSelectedFile(new File(msg.content));

int returnVal = jf.showSaveDialog(ui);

String saveTo = jf.getSelectedFile().getPath();

if(saveTo != null && returnVal == JFileChooser.APPROVE\_OPTION){

Download dwn = new Download(saveTo, ui);

Thread t = new Thread(dwn);

t.start();

//send(new Message("upload\_res", (""+InetAddress.getLocalHost().getHostAddress()), (""+dwn.port), msg.sender));

send(new Message("upload\_res", ui.username, (""+dwn.port), msg.sender));

}

else{

send(new Message("upload\_res", ui.username, "NO", msg.sender));

}

}

else{

send(new Message("upload\_res", ui.username, "NO", msg.sender));

}

}

else if(msg.type.equals("upload\_res")){

if(!msg.content.equals("NO")){

int port = Integer.parseInt(msg.content);

String addr = msg.sender;

ui.jButton5.setEnabled(false); ui.jButton6.setEnabled(false);

Upload upl = new Upload(addr, port, ui.file, ui);

Thread t = new Thread(upl);

t.start();

}

else{

ui.jTextArea1.append("[SERVER > Me] : "+msg.sender+" rejected file request\n");

}

}

else{

ui.jTextArea1.append("[SERVER > Me] : Unknown message type\n");

}

}

catch(Exception ex) {

keepRunning = false;

ui.jTextArea1.append("[Application > Me] : Connection Failure\n");

ui.jButton1.setEnabled(true); ui.jTextField1.setEditable(true); ui.jTextField2.setEditable(true);

ui.jButton4.setEnabled(false); ui.jButton5.setEnabled(false); ui.jButton5.setEnabled(false);

for(int i = 1; i < ui.model.size(); i++){

ui.model.removeElementAt(i);

}

ui.clientThread.stop();

System.out.println("Exception SocketClient run()");

ex.printStackTrace();

}

}

}

public void send(Message msg){

try {

Out.writeObject(msg);

Out.flush();

System.out.println("Outgoing : "+msg.toString());

if(msg.type.equals("message") && !msg.content.equals(".bye")){

String msgTime = (new Date()).toString();

try{

hist.addMessage(msg, msgTime);

DefaultTableModel table = (DefaultTableModel) ui.historyFrame.jTable1.getModel();

table.addRow(new Object[]{"Me", msg.content, msg.recipient, msgTime});

}

catch(Exception ex){}

}

}

catch (IOException ex) {

System.out.println("Exception SocketClient send()");

}

}

public void closeThread(Thread t){

t = null;

}

}

**Upload.java**: This class initiates the file transfer protocol and sends the file in buffers as soon as the connection is accepted from the other end.

package com.socket;

import com.ui.ChatFrame;

import java.io.\*;

import java.net.\*;

import java.util.logging.Level;

import java.util.logging.Logger;

public class Upload implements Runnable{

public String addr;

public int port;

public Socket socket;

public FileInputStream In;

public OutputStream Out;

public File file;

public ChatFrame ui;

public Upload(String addr, int port, File filepath, ChatFrame frame){

super();

try {

file = filepath; ui = frame;

socket = new Socket(InetAddress.getByName(addr), port);

Out = socket.getOutputStream();

In = new FileInputStream(filepath);

}

catch (Exception ex) {

System.out.println("Exception [Upload : Upload(...)]");

}

}

@Override

public void run() {

try {

byte[] buffer = new byte[1024];

int count;

while((count = In.read(buffer)) >= 0){

Out.write(buffer, 0, count);

}

Out.flush();

ui.jTextArea1.append("[Applcation > Me] : File upload complete\n");

ui.jButton5.setEnabled(true); ui.jButton6.setEnabled(true);

ui.jTextField5.setVisible(true);

if(In != null){ In.close(); }

if(Out != null){ Out.close(); }

if(socket != null){ socket.close(); }

}

catch (Exception ex) {

System.out.println("Exception [Upload : run()]");

ex.printStackTrace();

}

}

}

* **com.ui**

1. ChatFrame.java
2. HistoryFrame.java

**ChatFrame.java**: This class draws the visible window, called frame for displaying user interface components like textfields, comboboxes, textareas, filechooser, panes, labels and many more. All for the chat.

package com.ui;

import com.socket.History;

import com.socket.Message;

import com.socket.SocketClient;

import java.awt.event.WindowEvent;

import java.awt.event.WindowListener;

import java.io.File;

import javax.swing.DefaultListModel;

import javax.swing.JFileChooser;

import javax.swing.JFrame;

import javax.swing.UIManager;

public class ChatFrame extends javax.swing.JFrame {

public SocketClient client;

public int port;

public String serverAddr, username, password;

public Thread clientThread;

public DefaultListModel model;

public File file;

public String historyFile = "D:/History.xml";

public HistoryFrame historyFrame;

public History hist;

public ChatFrame() {

initComponents();

this.setTitle("jMessenger");

model.addElement("All");

jList1.setSelectedIndex(0);

jTextField6.setEditable(false);

this.addWindowListener(new WindowListener() {

@Override public void windowOpened(WindowEvent e) {}

@Override public void windowClosing(WindowEvent e) { try{ client.send(new Message("message", username, ".bye", "SERVER")); clientThread.stop(); }catch(Exception ex){} }

@Override public void windowClosed(WindowEvent e) {}

@Override public void windowIconified(WindowEvent e) {}

@Override public void windowDeiconified(WindowEvent e) {}

@Override public void windowActivated(WindowEvent e) {}

@Override public void windowDeactivated(WindowEvent e) {}

});

hist = new History(historyFile);

}

public boolean isWin32(){

return System.getProperty("os.name").startsWith("Windows");

}

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents

private void initComponents() {

jLabel1 = new javax.swing.JLabel();

jTextField1 = new javax.swing.JTextField();

jLabel2 = new javax.swing.JLabel();

jTextField2 = new javax.swing.JTextField();

jButton1 = new javax.swing.JButton();

jTextField3 = new javax.swing.JTextField();

jLabel3 = new javax.swing.JLabel();

jLabel4 = new javax.swing.JLabel();

jButton3 = new javax.swing.JButton();

jPasswordField1 = new javax.swing.JPasswordField();

jSeparator1 = new javax.swing.JSeparator();

jScrollPane1 = new javax.swing.JScrollPane();

jTextArea1 = new javax.swing.JTextArea();

jScrollPane2 = new javax.swing.JScrollPane();

jList1 = new javax.swing.JList();

jLabel5 = new javax.swing.JLabel();

jTextField4 = new javax.swing.JTextField();

jButton4 = new javax.swing.JButton();

jButton2 = new javax.swing.JButton();

jSeparator2 = new javax.swing.JSeparator();

jTextField5 = new javax.swing.JTextField();

jButton5 = new javax.swing.JButton();

jButton6 = new javax.swing.JButton();

jLabel6 = new javax.swing.JLabel();

jLabel7 = new javax.swing.JLabel();

jTextField6 = new javax.swing.JTextField();

jButton7 = new javax.swing.JButton();

jButton8 = new javax.swing.JButton();

jLabel8 = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

jLabel1.setText("Server IP:");

jTextField1.setText("localhost");

jLabel2.setText("Port : ");

jTextField2.setText("13000");

jButton1.setText("Connect");

jButton1.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton1ActionPerformed(evt);

}

});

jTextField3.setText("Daniel");

jTextField3.setEnabled(false);

jLabel3.setText("Password :");

jLabel4.setText("Username :");

jButton3.setText("SignUp");

jButton3.setEnabled(false);

jButton3.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton3ActionPerformed(evt);

}

});

jPasswordField1.setText("1234");

jPasswordField1.setEnabled(false);

jTextArea1.setColumns(20);

jTextArea1.setFont(new java.awt.Font("Consolas", 0, 12)); // NOI18N

jTextArea1.setRows(5);

jScrollPane1.setViewportView(jTextArea1);

jList1.setModel((model = new DefaultListModel()));

jScrollPane2.setViewportView(jList1);

jLabel5.setText("Message : ");

jButton4.setText("Send Message ");

jButton4.setEnabled(false);

jButton4.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton4ActionPerformed(evt);

}

});

jButton2.setText("Login");

jButton2.setEnabled(false);

jButton2.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton2ActionPerformed(evt);

}

});

jButton5.setText("...");

jButton5.setEnabled(false);

jButton5.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton5ActionPerformed(evt);

}

});

jButton6.setText("Send");

jButton6.setEnabled(false);

jButton6.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton6ActionPerformed(evt);

}

});

jLabel6.setText("File :");

jLabel7.setText("History File :");

jButton7.setText("...");

jButton7.setEnabled(false);

jButton7.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton7ActionPerformed(evt);

}

});

jButton8.setText("Show");

jButton8.setEnabled(false);

jButton8.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton8ActionPerformed(evt);

}

});

jLabel8.setText("Conversation Window");

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jSeparator1, javax.swing.GroupLayout.Alignment.TRAILING)

.addComponent(jSeparator2)

.addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)

.addComponent(jLabel1)

.addComponent(jLabel4)

.addComponent(jLabel7))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jTextField3)

.addComponent(jTextField1))

.addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)

.addComponent(jLabel2)

.addComponent(jLabel3))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jTextField2)

.addComponent(jPasswordField1)))

.addComponent(jTextField6))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addComponent(jButton1, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)

.addComponent(jButton2, javax.swing.GroupLayout.PREFERRED\_SIZE, 70, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jButton7, javax.swing.GroupLayout.PREFERRED\_SIZE, 70, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addComponent(jButton8, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jButton3, javax.swing.GroupLayout.DEFAULT\_SIZE, 81, Short.MAX\_VALUE)))))

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()

.addComponent(jScrollPane1)

.addGap(18, 18, 18)

.addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED\_SIZE, 108, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()

.addComponent(jLabel5)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(jTextField4)

.addGap(18, 18, 18)

.addComponent(jButton4, javax.swing.GroupLayout.PREFERRED\_SIZE, 108, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jLabel8)

.addGroup(layout.createSequentialGroup()

.addGap(23, 23, 23)

.addComponent(jLabel6)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addComponent(jTextField5, javax.swing.GroupLayout.PREFERRED\_SIZE, 378, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(18, 18, 18)

.addComponent(jButton5, javax.swing.GroupLayout.PREFERRED\_SIZE, 27, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(jButton6, javax.swing.GroupLayout.PREFERRED\_SIZE, 77, javax.swing.GroupLayout.PREFERRED\_SIZE)))

.addGap(0, 0, Short.MAX\_VALUE)))

.addContainerGap())

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jLabel1)

.addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabel2)

.addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jButton1))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jTextField3, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabel3)

.addComponent(jLabel4)

.addComponent(jButton3)

.addComponent(jPasswordField1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jButton2))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jLabel7)

.addComponent(jTextField6, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jButton7)

.addComponent(jButton8))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addComponent(jSeparator1, javax.swing.GroupLayout.PREFERRED\_SIZE, 12, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jLabel8)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addComponent(jScrollPane1)

.addComponent(jScrollPane2, javax.swing.GroupLayout.DEFAULT\_SIZE, 264, Short.MAX\_VALUE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jButton4)

.addComponent(jLabel5)

.addComponent(jTextField4, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addComponent(jSeparator2, javax.swing.GroupLayout.PREFERRED\_SIZE, 10, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE, false)

.addComponent(jButton6, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jButton5, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jLabel6)

.addComponent(jTextField5, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addContainerGap())

);

pack();

}// </editor-fold>//GEN-END:initComponents

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_jButton1ActionPerformed

serverAddr = jTextField1.getText(); port = Integer.parseInt(jTextField2.getText());

if(!serverAddr.isEmpty() && !jTextField2.getText().isEmpty()){

try{

client = new SocketClient(this);

clientThread = new Thread(client);

clientThread.start();

client.send(new Message("test", "testUser", "testContent", "SERVER"));

}

catch(Exception ex){

jTextArea1.append("[Application > Me] : Server not found\n");

}

}

}//GEN-LAST:event\_jButton1ActionPerformed

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_jButton2ActionPerformed

username = jTextField3.getText();

password = jPasswordField1.getText();

if(!username.isEmpty() && !password.isEmpty()){

client.send(new Message("login", username, password, "SERVER"));

}

}//GEN-LAST:event\_jButton2ActionPerformed

private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_jButton4ActionPerformed

String msg = jTextField4.getText();

String target = jList1.getSelectedValue().toString();

if(!msg.isEmpty() && !target.isEmpty()){

jTextField4.setText("");

client.send(new Message("message", username, msg, target));

}

}//GEN-LAST:event\_jButton4ActionPerformed

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_jButton3ActionPerformed

username = jTextField3.getText();

password = jPasswordField1.getText();

if(!username.isEmpty() && !password.isEmpty()){

client.send(new Message("signup", username, password, "SERVER"));

}

}//GEN-LAST:event\_jButton3ActionPerformed

private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_jButton5ActionPerformed

JFileChooser fileChooser = new JFileChooser();

fileChooser.showDialog(this, "Select File");

file = fileChooser.getSelectedFile();

if(file != null){

if(!file.getName().isEmpty()){

jButton6.setEnabled(true); String str;

if(jTextField5.getText().length() > 30){

String t = file.getPath();

str = t.substring(0, 20) + " [...] " + t.substring(t.length() - 20, t.length());

}

else{

str = file.getPath();

}

jTextField5.setText(str);

}

}

}//GEN-LAST:event\_jButton5ActionPerformed

private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_jButton6ActionPerformed

long size = file.length();

if(size < 120 \* 1024 \* 1024){

client.send(new Message("upload\_req", username, file.getName(), jList1.getSelectedValue().toString()));

}

else{

jTextArea1.append("[Application > Me] : File is size too large\n");

}

}//GEN-LAST:event\_jButton6ActionPerformed

private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_jButton7ActionPerformed

JFileChooser jf = new JFileChooser();

jf.showDialog(this, "Select File");

if(!jf.getSelectedFile().getPath().isEmpty()){

historyFile = jf.getSelectedFile().getPath();

if(this.isWin32()){

historyFile = historyFile.replace("/", "\\");

}

jTextField6.setText(historyFile);

jTextField6.setEditable(false);

jButton7.setEnabled(false);

jButton8.setEnabled(true);

hist = new History(historyFile);

historyFrame = new HistoryFrame(hist);

historyFrame.setDefaultCloseOperation(JFrame.HIDE\_ON\_CLOSE);

historyFrame.setVisible(false);

}

}//GEN-LAST:event\_jButton7ActionPerformed

private void jButton8ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_jButton8ActionPerformed

historyFrame.setLocation(this.getLocation());

historyFrame.setVisible(true);

}//GEN-LAST:event\_jButton8ActionPerformed

public static void main(String args[]) {

try {

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());

}

catch(Exception ex){

System.out.println("Look & Feel exception");

}

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

new ChatFrame().setVisible(true);

}

});

}

// Variables declaration - do not modify//GEN-BEGIN:variables

public javax.swing.JButton jButton1;

public javax.swing.JButton jButton2;

public javax.swing.JButton jButton3;

public javax.swing.JButton jButton4;

public javax.swing.JButton jButton5;

public javax.swing.JButton jButton6;

public javax.swing.JButton jButton7;

public javax.swing.JButton jButton8;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private javax.swing.JLabel jLabel3;

private javax.swing.JLabel jLabel4;

private javax.swing.JLabel jLabel5;

private javax.swing.JLabel jLabel6;

private javax.swing.JLabel jLabel7;

private javax.swing.JLabel jLabel8;

public javax.swing.JList jList1;

public javax.swing.JPasswordField jPasswordField1;

private javax.swing.JScrollPane jScrollPane1;

private javax.swing.JScrollPane jScrollPane2;

private javax.swing.JSeparator jSeparator1;

private javax.swing.JSeparator jSeparator2;

public javax.swing.JTextArea jTextArea1;

public javax.swing.JTextField jTextField1;

public javax.swing.JTextField jTextField2;

public javax.swing.JTextField jTextField3;

public javax.swing.JTextField jTextField4;

public javax.swing.JTextField jTextField5;

public javax.swing.JTextField jTextField6;

// End of variables declaration//GEN-END:variables

}

**HistoryFrame.java**: This class also draws the visible user interface just like the ChatFrame.java class but this handles displaying the history instead.

package com.ui;

import com.socket.History;

public class HistoryFrame extends javax.swing.JFrame {

public History hist;

public HistoryFrame() {

initComponents();

}

public HistoryFrame(History hist){

initComponents();

this.hist = hist;

hist.FillTable(this);

}

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents

private void initComponents() {

jLabel1 = new javax.swing.JLabel();

jScrollPane1 = new javax.swing.JScrollPane();

jTable1 = new javax.swing.JTable();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

setTitle("Chat History");

jLabel1.setText("History : ");

jTable1.setModel(new javax.swing.table.DefaultTableModel(

new Object [][] {

},

new String [] {

"Sender", "Message", "To", "Time"

}

) {

Class[] types = new Class [] {

java.lang.String.class, java.lang.String.class, java.lang.String.class, java.lang.String.class

};

boolean[] canEdit = new boolean [] {

false, false, false, false

};

public Class getColumnClass(int columnIndex) {

return types [columnIndex];

}

public boolean isCellEditable(int rowIndex, int columnIndex) {

return canEdit [columnIndex];

}

});

jScrollPane1.setViewportView(jTable1);

jTable1.getColumnModel().getColumn(1).setPreferredWidth(200);

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addComponent(jLabel1)

.addGap(0, 0, Short.MAX\_VALUE))

.addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT\_SIZE, 485, Short.MAX\_VALUE))

.addContainerGap())

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addComponent(jLabel1)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT\_SIZE, 289, Short.MAX\_VALUE)

.addContainerGap())

);

pack();

}// </editor-fold>//GEN-END:initComponents

public static void main(String args[]) {

/\* Set the Nimbus look and feel \*/

//<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">

/\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.

\* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html

\*/

try {

for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {

if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());

break;

}

}

} catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(HistoryFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(HistoryFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(HistoryFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(HistoryFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

}

//</editor-fold>

/\* Create and display the form \*/

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

new HistoryFrame().setVisible(true);

}

});

}

// Variables declaration - do not modify//GEN-BEGIN:variables

private javax.swing.JLabel jLabel1;

private javax.swing.JScrollPane jScrollPane1;

public javax.swing.JTable jTable1;

// End of variables declaration//GEN-END:variables

}