

5 de febrero de 2018

Ruben Palomo Yuste

D.A.M

Los Naranjos

Chat Con Sockets MeNESAJE ENCRIPTADOS

Chat creado con sockets e hilos

***Índice:***

1. ***Cliente.***
   1. ***ThreadRecibeCliente.***
   2. ***ThreadEnviaCliente.***
   3. ***VistaCliente.***
2. ***Servidor.***
   1. ***Mensajes.***
   2. ***ThreadEnviaServidor.***
   3. ***ThreadRecibeServidor.***
   4. ***VistaServidor.***
3. ***Cliente.***
   1. ***ThreadRecibeCliente.***

package Cliente;

import java.io.DataInputStream;

import java.io.EOFException;

import java.io.IOException;

import java.net.Socket;

import java.net.SocketException;

import java.util.logging.Level;

import java.util.logging.Logger;

import java.security.MessageDigest;

import java.util.Arrays;

import javax.crypto.Cipher;

import javax.crypto.SecretKey;

import javax.crypto.spec.SecretKeySpec;

import org.apache.commons.codec.binary.Base64;

public class ThreadRecibeCliente implements Runnable {

private final VistaCliente main; // Frame del cliente

private String mensaje; // Mensaje que recibe el cliente

private DataInputStream entrada; // Flujo de entrada de objetos

private final Socket cliente; // Socket del cliente

/\*\*

\* Constructor vacio

\*/

public ThreadRecibeCliente(){

this.cliente = null;

this.main = null;

}

/\*\*

\* Constructor

\* @param cliente

\* @param main

\*/

public ThreadRecibeCliente(Socket cliente, VistaCliente main){

this.cliente = cliente;

this.main = main;

}

@Override

public void run() {

try {

// Abrimos un flujo de entrada de objetos

entrada = new DataInputStream(cliente.getInputStream());

} catch (IOException ex) {

Logger.getLogger(ThreadRecibeCliente.class.getName()).log(Level.SEVERE, null, ex);

}

// Recorre hasta que se cierra el cliente

do {

// Intenta

try {

// Leer un nuevo mensaje

mensaje = (String) entrada.readUTF();

mensaje = desencriptar(mensaje);

// Llama al metodo que lo muestra y se lo pasa por parametro

main.mostrarMensaje(mensaje);

}catch(SocketException | EOFException ex){

try {

// Cerramos la ventana

System.exit(0);

// Cerramos el cliente

cliente.close();

} catch (IOException ex1) {

Logger.getLogger(ThreadRecibeCliente.class.getName()).log(Level.SEVERE, null, ex1);

}

}catch (IOException ex) {

Logger.getLogger(ThreadRecibeCliente.class.getName()).log(Level.SEVERE, null, ex);

}

} while (true); //Ejecuta hasta que el server escriba TERMINATE

}

/\*\*

\* Metodo para desencriptar

\* @param secretKey

\* @return

\*/

public String desencriptar(String desencriptado) {

String secretKey="ey esta es la clave";

String base64EncryptedString = "";

try {

byte[] message = Base64.decodeBase64(desencriptado.getBytes("utf-8"));

MessageDigest md = MessageDigest.getInstance("MD5");

byte[] digestOfPassword = md.digest(secretKey.getBytes("utf-8"));

byte[] keyBytes = Arrays.copyOf(digestOfPassword, 24);

SecretKey key = new SecretKeySpec(keyBytes, "DESede");

Cipher decipher = Cipher.getInstance("DESede");

decipher.init(Cipher.DECRYPT\_MODE, key);

byte[] plainText = decipher.doFinal(message);

base64EncryptedString = new String(plainText, "UTF-8");

} catch (Exception ex) {

}

System.out.println(base64EncryptedString);

return base64EncryptedString;

}

}

* 1. ***ThreadEnviaCliente.***

package Cliente;

import java.io.IOException;

import java.net.Socket;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.DataOutputStream;

import java.security.MessageDigest;

import java.util.Arrays;

import javax.crypto.Cipher;

import javax.crypto.SecretKey;

import javax.crypto.spec.SecretKeySpec;

import org.apache.commons.codec.binary.Base64;

public class ThreadEnviaCliente implements Runnable {

private final VistaCliente main; // Frame del cliente

private DataOutputStream salida; // Flujo de salida de datos

private String mensaje; // Mensaje para enviar

private final Socket cliente; // Socket de Cliente

/\*\*

\* Constructor vacio

\*/

public ThreadEnviaCliente(){

this.cliente = null;

this.main = null;

}

/\*\*

\* Constructor

\* @param cliente

\* @param main

\*/

public ThreadEnviaCliente(Socket cliente, final VistaCliente main){

this.cliente = cliente;

this.main = main;

/\*\*

\* Evento que ocurre cuando se pulsa el boton enviar

\*/

VistaCliente.btnEnviar.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent event) {

// Cogemos el texto que hay dentro del txtField

mensaje = VistaCliente.txtEnviar.getText();

mensaje = encriptar(mensaje);

// Se envia el mensaje

enviarDatos(mensaje);

// Borra el texto del TextView

VistaCliente.txtEnviar.setText("");

}

});

}

@Override

public void run() {

try {

// Abrimos un flujo de salida de objetos hacia el servidor

salida = new DataOutputStream(cliente.getOutputStream());

} catch (IOException e) {

System.out.println("ERROR: " + e);

}

}

/\*\*

\* Envia los datos al servidor

\* @param mensaje

\*/

private void enviarDatos(String mensaje){

try {

// Escribe el mensaje que le pasan por parametro

salida.writeUTF(mensaje);

// Flush salida a servidor

salida.flush();

}catch (IOException e){

System.out.println("Error escribiendo Mensaje");

}

}

public String encriptar(String encriptado) {

String keySecret="ey esta es la clave";

String base64EncryptedString = "";

try {

MessageDigest md = MessageDigest.getInstance("MD5");

byte[] digestOfPassword = md.digest(keySecret.getBytes("utf-8"));

byte[] keyBytes = Arrays.copyOf(digestOfPassword, 24);

SecretKey key = new SecretKeySpec(keyBytes, "DESede");

Cipher cipher = Cipher.getInstance("DESede");

cipher.init(Cipher.ENCRYPT\_MODE, key);

byte[] plainTextBytes = encriptado.getBytes("utf-8");

byte[] buf = cipher.doFinal(plainTextBytes);

byte[] base64Bytes = Base64.encodeBase64(buf);

base64EncryptedString = new String(base64Bytes);

} catch (Exception ex) {

}

System.out.println(base64EncryptedString);

return base64EncryptedString;

}

}

* 1. ***VistaCliente.***

package Cliente;

import java.io.IOException;

import java.net.ConnectException;

import java.net.Socket;

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.swing.JFrame;

/\*\*

\*

\* @author fp

\*/

public class VistaCliente extends javax.swing.JFrame {

private static Socket cliente; //Socket para conectarse con el cliente

private static String host; //ip a la cual se conecta

private static int puerto; //puerto a la cual se conecta

/\*\*

\* Creates new form VistaCliente

\*/

public VistaCliente() {

// Establece titulo a la Vista

super("Cliente");

initComponents();

setVisible(true);

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

jScrollPane1 = new javax.swing.JScrollPane();

txtMensajesAceptados = new javax.swing.JTextArea();

jLabel1 = new javax.swing.JLabel();

jLabel2 = new javax.swing.JLabel();

jScrollPane2 = new javax.swing.JScrollPane();

txtMensajesRechazados = new javax.swing.JTextArea();

jLabel3 = new javax.swing.JLabel();

btnEnviar = new javax.swing.JButton();

txtEnviar = new javax.swing.JTextField();

lblErrores = new javax.swing.JLabel();

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

txtMensajesAceptados.setColumns(20);

txtMensajesAceptados.setRows(5);

jScrollPane1.setViewportView(txtMensajesAceptados);

jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);

jLabel1.setText("Mensajes Aceptados");

jLabel1.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(0, 0, 0)));

jLabel2.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);

jLabel2.setText("Mensajes Rechazar");

jLabel2.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(0, 0, 0)));

txtMensajesRechazados.setColumns(20);

txtMensajesRechazados.setRows(5);

jScrollPane2.setViewportView(txtMensajesRechazados);

jLabel3.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);

jLabel3.setText("Escribe tu mensaje:");

jLabel3.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(0, 0, 0)));

btnEnviar.setText("Enviar");

lblErrores.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);

lblErrores.setText("....");

lblErrores.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(0, 0, 0)));

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.TRAILING, false)

.addGroup(layout.createSequentialGroup()

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

.addGap(4, 4, 4))

.addComponent(jLabel3, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

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

.addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED\_SIZE, 280, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, 280, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED\_SIZE, 280, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGroup(layout.createSequentialGroup()

.addComponent(txtEnviar, javax.swing.GroupLayout.PREFERRED\_SIZE, 196, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(18, 18, 18)

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

.addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED\_SIZE, 280, javax.swing.GroupLayout.PREFERRED\_SIZE)))

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

);

layout.setVerticalGroup(

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

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addComponent(jLabel3)

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

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

.addComponent(btnEnviar)

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

.addGap(18, 18, 18)

.addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED\_SIZE, 30, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, 101, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED\_SIZE, 30, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED\_SIZE, 101, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addComponent(lblErrores, javax.swing.GroupLayout.PREFERRED\_SIZE, 33, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

);

pack();

}// </editor-fold>

/\*\*

\* @param args the command line arguments

\*/

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(VistaCliente.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

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

} catch (IllegalAccessException ex) {

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

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

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

}

//</editor-fold>

// Instanciacion de la clase VistaCliente

VistaCliente main = new VistaCliente();

// Centra el JFrame

main.setLocationRelativeTo(null);

// Habilita cerrar la ventana

main.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

// Para correr los threads

ExecutorService executor = Executors.newCachedThreadPool();

// Inicializamos el puerto y el host

puerto = 2444;

host = "localhost";

try {

// Comunicarme con el servidor

cliente = new Socket(host, puerto);

// Ejecucion de los Threads del cliente

executor.execute(new ThreadRecibeCliente(cliente, main));

executor.execute(new ThreadEnviaCliente(cliente, main));

}catch(ConnectException e){

lblErrores.setText("Servidor offline, volvemos pronto");

}catch (IOException ex) {

Logger.getLogger(VistaCliente.class.getName()).log(Level.SEVERE, null, ex);

}finally {

// Cerramos la ejecucion de los hilos

executor.shutdown();

}

}

// Variables declaration - do not modify

public static javax.swing.JButton btnEnviar;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private javax.swing.JLabel jLabel3;

private javax.swing.JScrollPane jScrollPane1;

private javax.swing.JScrollPane jScrollPane2;

private static javax.swing.JLabel lblErrores;

public static javax.swing.JTextField txtEnviar;

public static javax.swing.JTextArea txtMensajesAceptados;

private javax.swing.JTextArea txtMensajesRechazados;

// End of variables declaration

/\*\*

\* Muestra el mensaje que recibe y se acepta o rechaza

\* @param mensaje

\*/

public void mostrarMensaje(String mensaje) {

// Si el mensaje empieza con validado

if(mensaje.startsWith("Validado")){

// Se pone en los validados

txtMensajesAceptados.append(mensaje + "\n");

}else{

// Se pone en los rechazados

txtMensajesRechazados.append(mensaje + "\n");

}

}

}

1. ***Servidor.***
   1. ***Mensajes.***

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package Servidor;

/\*\*

\*

\* @author fp

\*/

public class Mensajes {

private String mensajeCliente;

public Mensajes(){

mensajeCliente = "";

}

public Mensajes(String mensajeCliente){

this.mensajeCliente = mensajeCliente;

}

public String getMensajeCliente(){

return mensajeCliente;

}

/\*\*

\* Paso a String

\* @return

\*/

@Override

public String toString(){

return ", Mensaje: " + mensajeCliente;

}

}

* 1. ***ThreadEnviaServidor.***

package Servidor;

import java.io.IOException;

import java.net.Socket;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.DataOutputStream;

import java.net.SocketException;

import java.security.MessageDigest;

import java.util.Arrays;

import javax.crypto.Cipher;

import javax.crypto.SecretKey;

import javax.crypto.spec.SecretKeySpec;

import org.apache.commons.codec.binary.Base64;

public class ThreadEnviaServidor implements Runnable {

private final VistaServidor main;

private DataOutputStream salida;

private String mensaje;

private final Socket conexion;

/\*\*

\* Constructor vacio

\*/

public ThreadEnviaServidor(){

this.conexion = null;

this.main = null;

}

/\*\*

\* Contructor

\* @param conexion

\* @param main

\*/

public ThreadEnviaServidor(Socket conexion, final VistaServidor main){

this.conexion = conexion;

this.main = main;

/\*\*

\* Evento que ocurre cuando se pulsa el boton validar

\*/

VistaServidor.btnValidar.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent event) {

// Crea el mensaje que va a enviar

mensaje = "Validado: " + VistaServidor.textArea2.getSelectedValue();

mensaje = encriptar(mensaje);

// Se llama al metodo que envia el mensaje

enviarDatos(mensaje);

}

});

/\*\*

\* Evento que ocurre cuando se pulsa el boton rechazar

\*/

VistaServidor.btnRechazar.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent event) {

// Crea el mensaje que va a enviar

mensaje = "Rechazado: " + VistaServidor.textArea2.getSelectedValue();

mensaje = encriptar(mensaje);

// Se llama al metodo que envia el mensaje

enviarDatos(mensaje);

}

});

}

@Override

public void run() {

try {

// Abro un flujo de salida de datos hacia el cliente

salida = new DataOutputStream(conexion.getOutputStream());

} catch (SocketException | NullPointerException ex) {

System.out.println("ERROR: " + ex);

} catch (IOException e) {

System.out.println("ERROR: " + e);

}

}

/\*\*

\* Metodo que envia el mensaje

\* @param mensaje

\*/

private void enviarDatos(String mensaje){

try {

// Escribe el mensaje

salida.writeUTF(mensaje);

// Salida de datos al cliente

salida.flush();

}catch (IOException e){

System.out.println("Error escribiendo Mensaje: " + e);

}

}

public String encriptar(String encriptado) {

String keySecret="ey esta es la clave";

String base64EncryptedString = "";

try {

MessageDigest md = MessageDigest.getInstance("MD5");

byte[] digestOfPassword = md.digest(keySecret.getBytes("utf-8"));

byte[] keyBytes = Arrays.copyOf(digestOfPassword, 24);

SecretKey key = new SecretKeySpec(keyBytes, "DESede");

Cipher cipher = Cipher.getInstance("DESede");

cipher.init(Cipher.ENCRYPT\_MODE, key);

byte[] plainTextBytes = encriptado.getBytes("utf-8");

byte[] buf = cipher.doFinal(plainTextBytes);

byte[] base64Bytes = Base64.encodeBase64(buf);

base64EncryptedString = new String(base64Bytes);

} catch (Exception ex) {

}

System.out.println(base64EncryptedString);

return base64EncryptedString;

}

}

* 1. ***ThreadRecibeServidor.***

package Servidor;

import java.io.DataInputStream;

import java.io.EOFException;

import java.io.IOException;

import java.net.Socket;

import java.security.MessageDigest;

import java.util.Arrays;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.crypto.Cipher;

import javax.crypto.SecretKey;

import javax.crypto.spec.SecretKeySpec;

import org.apache.commons.codec.binary.Base64;

public class ThreadRecibeServidor implements Runnable {

private final VistaServidor main; // Frame del servidor

private String mensaje; // Mensaje que recibe

private DataInputStream entrada; // Flujo de entrada de objetos

private final Socket cliente; // Socket del cliente

/\*\*

\* Constructor vacio

\*/

public ThreadRecibeServidor(){

this.cliente = null;

this.main = null;

}

/\*\*

\* Constructor

\* @param cliente

\* @param main

\*/

public ThreadRecibeServidor(Socket cliente, VistaServidor main){

this.cliente = cliente;

this.main = main;

}

@Override

public void run() {

try {

// Se abre un flujo de entrada de datos

entrada = new DataInputStream(cliente.getInputStream());

} catch (IOException ex) {

Logger.getLogger(ThreadRecibeServidor.class.getName()).log(Level.SEVERE, null, ex);

}

// Recorre hasta que se cierra el servidor

do {

// Intenta

try {

// Lee un nuevo mensaje

mensaje = (String) entrada.readUTF();

mensaje = desencriptar(mensaje);

// Llama al metodo que te muestra el mensaje

main.mostrarMensaje(mensaje);

}catch (EOFException ex) {

System.out.println("Fin de la conexion: " + ex);

}catch (IOException ex) {

Logger.getLogger(ThreadRecibeServidor.class.getName()).log(Level.SEVERE, null, ex);

}

} while (true);

}

/\*\*

\* Metodo para desencriptar

\* @param secretKey

\* @return

\*/

public String desencriptar(String desencriptado) {

String secretKey="ey esta es la clave";

String base64EncryptedString = "";

try {

byte[] message = Base64.decodeBase64(desencriptado.getBytes("utf-8"));

MessageDigest md = MessageDigest.getInstance("MD5");

byte[] digestOfPassword = md.digest(secretKey.getBytes("utf-8"));

byte[] keyBytes = Arrays.copyOf(digestOfPassword, 24);

SecretKey key = new SecretKeySpec(keyBytes, "DESede");

Cipher decipher = Cipher.getInstance("DESede");

decipher.init(Cipher.DECRYPT\_MODE, key);

byte[] plainText = decipher.doFinal(message);

base64EncryptedString = new String(plainText, "UTF-8");

} catch (Exception ex) {

}

System.out.println(base64EncryptedString);

return base64EncryptedString;

}

}

* 1. ***VistaServidor.***

package Servidor;

import java.io.IOException;

import java.net.ServerSocket;

import java.net.Socket;

import java.util.ArrayList;

import java.util.Iterator;

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.swing.DefaultListModel;

import javax.swing.JFrame;

/\*\*

\*

\* @author fp

\*/

public class VistaServidor extends javax.swing.JFrame {

private static ServerSocket servidor; // Socket del servidor

private static Socket cliente; // Socket del cliente

private static int puerto; // Puerto a la cual se conecta

/\*\*

\* Creates new form VistaServidor

\*/

public VistaServidor() {

// Establece titulo a la Vista

super("Servidor");

initComponents();

// Inicializamos el arraylist

mensajes = new ArrayList();

setVisible(true);

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

jPanel1 = new javax.swing.JPanel();

jPanel2 = new javax.swing.JPanel();

jLabel1 = new javax.swing.JLabel();

btnValidar = new javax.swing.JButton();

btnRechazar = new javax.swing.JButton();

jScrollPane1 = new javax.swing.JScrollPane();

textArea2 = new javax.swing.JList<>();

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

jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);

jLabel1.setText("Mensaje por validar: ");

jLabel1.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(0, 0, 0)));

btnValidar.setText("Validar");

btnRechazar.setText("Rechazar");

jScrollPane1.setViewportView(textArea2);

javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);

jPanel2.setLayout(jPanel2Layout);

jPanel2Layout.setHorizontalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel2Layout.createSequentialGroup()

.addContainerGap()

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel2Layout.createSequentialGroup()

.addComponent(btnValidar, javax.swing.GroupLayout.PREFERRED\_SIZE, 130, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addComponent(btnRechazar, javax.swing.GroupLayout.PREFERRED\_SIZE, 138, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(jPanel2Layout.createSequentialGroup()

.addGap(87, 87, 87)

.addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED\_SIZE, 203, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(0, 83, Short.MAX\_VALUE))

.addComponent(jScrollPane1, javax.swing.GroupLayout.Alignment.TRAILING))

.addContainerGap())

);

jPanel2Layout.setVerticalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel2Layout.createSequentialGroup()

.addContainerGap()

.addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED\_SIZE, 29, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(11, 11, 11)

.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(btnRechazar, javax.swing.GroupLayout.DEFAULT\_SIZE, 55, Short.MAX\_VALUE)

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

.addContainerGap())

);

javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);

jPanel1.setLayout(jPanel1Layout);

jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jPanel2, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

);

jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

);

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

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

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

.addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

);

layout.setVerticalGroup(

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

.addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

);

pack();

}// </editor-fold>

/\*\*

\* @param args the command line arguments

\*/

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 | InstantiationException | IllegalAccessException | javax.swing.UnsupportedLookAndFeelException ex) {

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

}

//</editor-fold>

//Instanciacion de la clase VistaServidor

VistaServidor main = new VistaServidor();

//Centrar el JFrame

main.setLocationRelativeTo(null);

// Habilita cerrar la ventana

main.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

// Para correr los threads

ExecutorService executor = Executors.newCachedThreadPool();

// Inicializamos el puerto

puerto = 2444;

try {

// Inicializamos un nuevo serverSocket

servidor = new ServerSocket(puerto);

// Bucle infinito para esperar conexiones de los clientes

// Hasta que se cierre el servidor

while (true){

try {

//Permite al servidor aceptar conexiones

cliente = servidor.accept();

//Ejecucion de los threads del servidor

executor.execute(new ThreadRecibeServidor(cliente, main));

executor.execute(new ThreadEnviaServidor(cliente, main));

} catch (IOException ex) {

Logger.getLogger(VistaServidor.class.getName()).log(Level.SEVERE, null, ex);

}

}

} catch (IOException ex) {

Logger.getLogger(VistaServidor.class.getName()).log(Level.SEVERE, null, ex);

}finally{

try {

// Cerramos el servidor y la conexion con el cliente

servidor.close();

cliente.close();

// Cerramos la ejecucion de los hilos

executor.shutdown();

} catch (IOException ex) {

Logger.getLogger(VistaServidor.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

// Variables declaration - do not modify

public static javax.swing.JButton btnRechazar;

public static javax.swing.JButton btnValidar;

private javax.swing.JLabel jLabel1;

private javax.swing.JPanel jPanel1;

private javax.swing.JPanel jPanel2;

private javax.swing.JScrollPane jScrollPane1;

public static javax.swing.JList<String> textArea2;

// End of variables declaration

static ArrayList<Mensajes> mensajes;

//Para mostrar texto en displayArea

public void mostrarMensaje(String mensaje) {

Mensajes men = new Mensajes(mensaje);

mensajes.add(men);

DefaultListModel modelo = new DefaultListModel();

//Leemos el mensaje del cliente

Iterator<Mensajes> iterador = mensajes.iterator();

Mensajes mensajeLeer;

// Recorro los equipos

while(iterador.hasNext()){

mensajeLeer = iterador.next();

System.out.println(mensajeLeer.getMensajeCliente());

String mensaje2 = mensajeLeer.getMensajeCliente();

modelo.addElement(mensaje2);

}

textArea2.setModel(modelo);

}

}