RELATORIO COM LINHAS DE CÓDIGO DO PROGRAMA.

LINHAS DE CODIGO CLIENTE.

package juan.sockets;

import javax.swing.\*;

import java.awt.event.\*;

import java.io.DataOutputStream;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.io.Serializable;

import java.net.\*;

public class Cliente {

public static void main(String[] args) {

// TODO Auto-generated method stub

MarcoCliente mimarco=new MarcoCliente();

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

}

}

class MarcoCliente extends JFrame{

public MarcoCliente(){

setBounds(600,300,280,350);

LaminaMarcoCliente milamina=new LaminaMarcoCliente();

add(milamina);

setVisible(true);

}

}

class LaminaMarcoCliente extends JPanel implements Runnable{

public LaminaMarcoCliente(){

nick=new JTextField(5);

add(nick);

JLabel texto=new JLabel("-CHAT-");

add(texto);

ip=new JTextField(8);

add(ip);

campochat=new JTextArea(12,20);

add(campochat);

campo1=new JTextField(20);

add(campo1);

miboton=new JButton("Enviar");

EnviaTexto mievento=new EnviaTexto();

miboton.addActionListener(mievento);

add(miboton);

Thread mihilo=new Thread(this);

mihilo.start();

}

private class EnviaTexto implements ActionListener{

@Override

public void actionPerformed(ActionEvent e) {

// TODO Auto-generated method stub

//System.out.println(campo1.getText());

try {

Socket misocket=new Socket("192.168.43.123",9999);

PaqueteEnvio datos= new PaqueteEnvio();

datos.setNick(nick.getText());

datos.setIp(ip.getText());

datos.setMensaje(campo1.getText());

ObjectOutputStream paquete\_datos=new ObjectOutputStream(misocket.getOutputStream());

paquete\_datos.writeObject(datos);

misocket.close();

/\*DataOutputStream flujo\_salida=new DataOutputStream(misocket.getOutputStream());

Flujo\_sal ida.writeUTF(campo1.getText());

flujo\_salida.close();\*/

} catch (UnknownHostException e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

} catch (IOException e1) {

// TODO Auto-generated catch block

System.out.println(e1.getMessage());

}

}

}

private JTextField campo1, nick, ip ;

private JTextArea campochat;

private JButton miboton;

@Override

public void run() {

// TODO Auto-generated method stub

try {

ServerSocket servidor\_cliente=new ServerSocket(9090);

Socket cliente;

PaqueteEnvio paqueteRecebido;

while(true) {

cliente=servidor\_cliente.accept();

ObjectInputStream flujoentrada=new ObjectInputStream(cliente.getInputStream());

paqueteRecebido=(PaqueteEnvio) flujoentrada.readObject();

campochat.append("\n" + paqueteRecebido.getNick() + ": " + paqueteRecebido.getMensaje());

}

} catch(Exception e) {

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

}

}

}

class PaqueteEnvio implements Serializable{

private String nick, ip, mensaje;

public String getNick() {

return nick;

}

public void setNick(String nick) {

this.nick = nick;

}

public String getIp() {

return ip;

}

public void setIp(String ip) {

this.ip = ip;

}

public String getMensaje() {

return mensaje;

}

public void setMensaje(String mensaje) {

this.mensaje = mensaje;

}

}

LINHAS DE CÓDIGO SERVIDOR.

package juan.sockets;

import javax.swing.\*;

import java.awt.\*;

import java.io.\*;

import java.net.ServerSocket;

import java.net.\*;

public class Servidor {

public static void main(String[] args) {

// TODO Auto-generated method stub

MarcoServidor mimarco=new MarcoServidor();

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

}

}

class MarcoServidor extends JFrame implements Runnable{

public MarcoServidor(){

setBounds(1200,300,280,350);

JPanel milamina= new JPanel();

milamina.setLayout(new BorderLayout());

areatexto=new JTextArea();

milamina.add(areatexto,BorderLayout.CENTER);

add(milamina);

setVisible(true);

Thread mihilo=new Thread(this);

mihilo.start();

}

@Override

public void run() {

// TODO Auto-generated method stub

//System.out.println("Estoy a la escucha");

try {

ServerSocket servidor=new ServerSocket(9999);

String nick, ip, mensaje;

PaqueteEnvio paquete\_recebido;

while(true){

0 Socket misocket=servidor.accept();

ObjectInputStream paquete\_datos=new ObjectInputStream(misocket.getInputStream());

paquete\_recebido=(PaqueteEnvio) paquete\_datos.readObject();

nick=paquete\_recebido.getNick();

ip=paquete\_recebido.getIp();

mensaje=paquete\_recebido.getMensaje();

/\*DataInputStream flujo\_entrada=new DataInputStream(misocket.getInputStream());

String mensaje\_texto=flujo\_entrada.readUTF();

areatexto.append("\n" + mensaje\_texto);\*/

areatexto.append("\n" + nick + ": " + mensaje + "para" + ip);

Socket enviaDestinatario=new Socket(ip,9090);

ObjectOutputStream paqueteReenvio=new ObjectOutputStream(enviaDestinatario.getOutputStream());

paqueteReenvio.writeObject(paquete\_recebido);

paqueteReenvio.close();

enviaDestinatario.close();

misocket.close();

}

} catch (IOException | ClassNotFoundException e) {

// TODO Auto-generated catch block

e.printStackTrace();

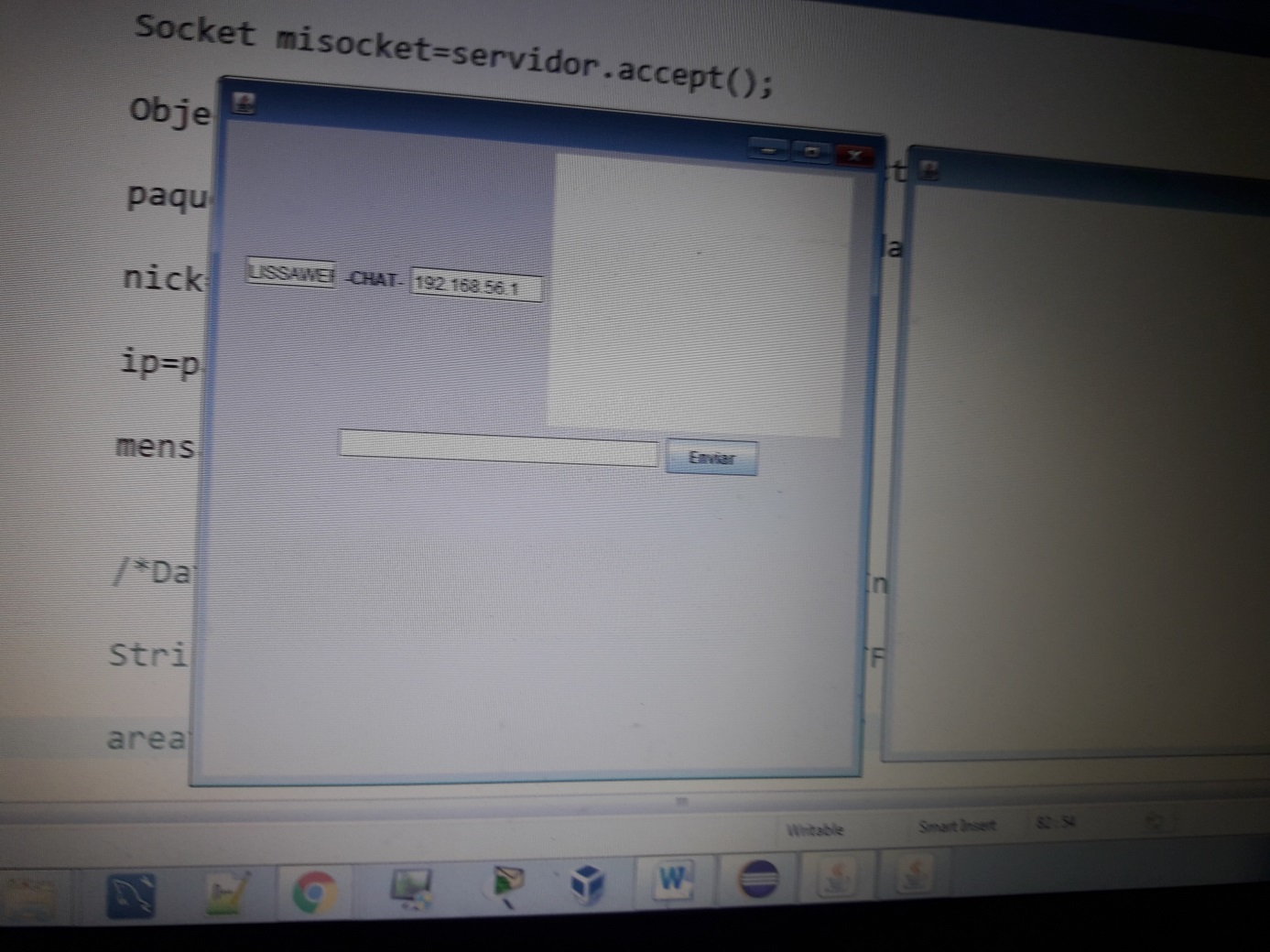
}

}

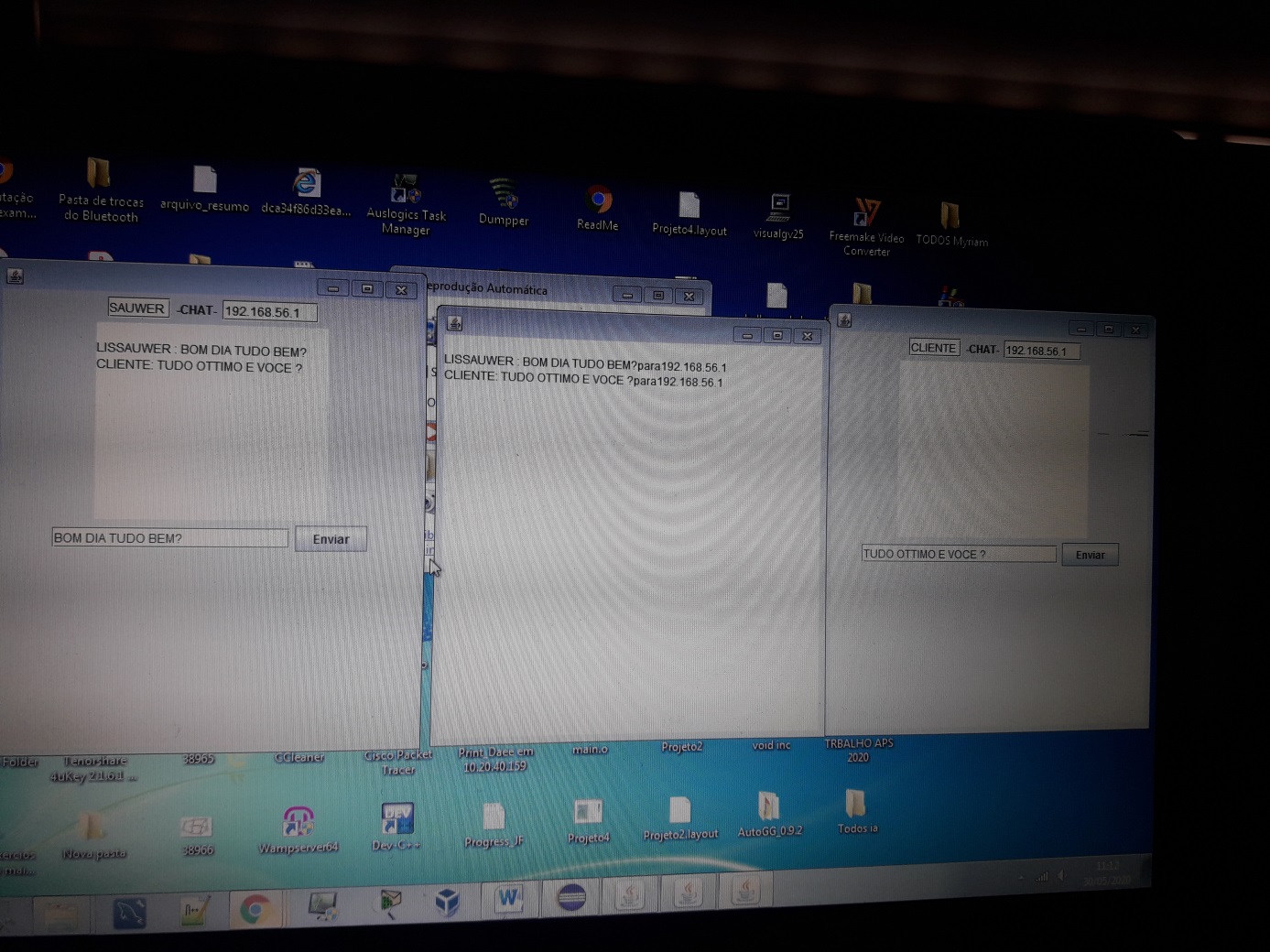
private JTextArea areatexto;

}

Principais janelas do programa em funcionamento e aguardando mensagens serem enviadas do Chat para o servidor.



Duplas de cliente e servidor em comunicção que sera remetida para qualquer o servidor definido pelo ip na rede.



Multiplas areas cliente servidores em comunicação, definindo a principal caracteritsica do Socket de possuir infinitas janelas.

