**4. Sarcina Problemei:**

**.** Se cere elaborarea unei perechi de programe client-server cu următoarea funcţionalitate:

-         serverul creează la prima conectare a unui client un fişier text vid cu numele acestuia (clientului);

-         clientul poate trimite serverului unul sau mai multe fişiere şi cere interclasarea acestora cu fişierul care-i poartă numele; rezultatul interclasării va fi reţinut în fişierul cu numele clientului;

-         clientul poate cere serverului returnarea conţinutului fişierului ce-i poartă numele.

**Listingul Programului:**

**Server:**

package server\_lab4;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.PrintWriter;

import java.net.ServerSocket;

import java.net.Socket;

public class Server\_Lab4 {

static int PORT, USERS;

public Server\_Lab4(int port, int users) throws IOException {

PORT = port;

USERS = users;

Socket[] mysocks = new Socket[USERS];

String[] names = new String[USERS];

Socket temp\_sock = null;

int count = 0;

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

mysocks[i] = null;

names[i] = null;

}

ServerSocket serverSocket = null;

boolean listening = true;

System.out.println("Încercați să vă conectați la server...");

try {

serverSocket = new ServerSocket(PORT);

System.out.println("Serverul este conectat!");

} catch (IOException e) {

System.err.println("Server nu răspuns la portul: " + PORT);

System.exit(-1);

}

while (listening) {

if (count < USERS && ((temp\_sock = serverSocket.accept()) != null)) {

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

if (mysocks[i] == null) {

count = i;

mysocks[i] = temp\_sock;

temp\_sock = null;

new ServerThread(mysocks[count], mysocks, count, names);

break;

}

}

count = 0;

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

if (mysocks[i] != null) {

count++;

}

}

System.out.println("Pe server este " + count + ".");

} else {

System.out.println("Serverul este plin");

}

}

serverSocket.close();

}

public static void main(String[] args) throws IOException {

new Server\_Lab4(8888, 10);

}

}

class ServerThread extends Thread {

String names[] = null;

String name = null;

public int own\_num;

private Socket socket = null;

private Socket[] mys = null;

PrintWriter outAll = null;

public ServerThread(Socket socket, Socket[] my, int num, String names[]) {

super("ServerThread");

this.socket = socket;

this.mys = my;

own\_num = num;

this.names = names;

this.start();

}

@Override

public void run() {

try {

PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

BufferedReader in = new BufferedReader(

new InputStreamReader(

socket.getInputStream()));

String inputLine, outputLine;

while ((inputLine = in.readLine()) != null) {

if (inputLine.equalsIgnoreCase("2112333")) {

for (int i = 0; i < Server\_Lab4.USERS; i++) {

if (names[i] != null) {

if (name.compareTo(names[i]) == 0) {

names[i] = null;

break;

}

}

}

for (int i = 0; i < Server\_Lab4.USERS; i++) {

if (mys[i] != null) {

outAll = new PrintWriter(mys[i].getOutputStream(), true);

outAll.println("^#");

for (int j = 0; j < Server\_Lab4.USERS; j++) {

if (names[j] != null) {

outAll.println(names[j]);

}

}

outAll = null;

}

}

break;

}

if (inputLine.charAt(0) == '^') {

for (int i = 0; i < Server\_Lab4.USERS; i++) {

if (names[i] == null) {

name = inputLine;

names[i] = inputLine;

break;

}

}

for (int i = 0; i < Server\_Lab4.USERS; i++) {

if (mys[i] != null) {

outAll = new PrintWriter(mys[i].getOutputStream(), true);

outAll.println("^#");

for (int j = 0; j < Server\_Lab4.USERS; j++) {

if (names[j] != null) {

outAll.println(names[j]);

}

}

outAll = null;

}

}

} else {

for (int i = 0; i < Server\_Lab4.USERS; i++) {

if (mys[i] != null) {

outAll = new PrintWriter(mys[i].getOutputStream(), true);

outAll.println(inputLine);

outAll = null;

}

}

}

}

socket.close();

mys[own\_num] = null;

out.close();

in.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

**Client:**

package client\_lab4;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

import java.io.\*;

import java.net.Socket;

import java.net.UnknownHostException;

import java.text.DateFormat;

import java.util.Date;

import javax.swing.JFileChooser;

import javax.swing.JFrame;

class Interface extends Frame {

Dialog form = null;

TextArea wnd\_msg = null, wnd\_names = null;

Button b\_send = null, b\_transfer = null, b\_view = null;

TextField to\_send = null;

public Interface(String window\_name) {

super(window\_name);

setFont(new Font("Comic", Font.BOLD, 12));

setBackground(Color.black);

wnd\_msg = new TextArea("", 14, 50, TextArea.SCROLLBARS\_VERTICAL\_ONLY);

wnd\_msg.setEditable(false);

wnd\_names = new TextArea("", 14, 10, TextArea.SCROLLBARS\_VERTICAL\_ONLY);

wnd\_names.setEditable(false);

to\_send = new TextField(50);

b\_view = new Button("Vizualizare jurnal");

b\_transfer = new Button("Încarcă");

b\_send = new Button("Trimite");

Panel for\_sends = new Panel();

Panel for\_wnds = new Panel();

for\_sends.add(b\_view);

for\_sends.add(b\_transfer);

for\_sends.add(to\_send);

for\_sends.add(b\_send);

for\_wnds.add(wnd\_names);

for\_wnds.add(wnd\_msg);

add(for\_sends, BorderLayout.SOUTH);

add(for\_wnds, BorderLayout.CENTER);

setSize(650, 350);

setLocationCentre(this);

setResizable(false);

setVisible(true);

b\_transfer.addActionListener(

new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

JFileChooser chooser = new JFileChooser();

chooser.setCurrentDirectory(new File("."));

chooser.setFileFilter(

new javax.swing.filechooser.FileFilter() {

@Override

public boolean accept(File f) {

return f.getName().toLowerCase().endsWith(".txt")

|| f.isDirectory();

}

@Override

public String getDescription() {

return "TXT files";

}

});

int request = chooser.showOpenDialog(new JFrame());

if (request == JFileChooser.APPROVE\_OPTION) {

System.out.println("File load: " + chooser.getSelectedFile().getAbsolutePath());

try {

new TransferData(chooser.getSelectedFile(), Form.file);

} catch (IOException ex) {

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

}

}

}

});

b\_view.addActionListener(

new ActionListener() {

@Override

public void actionPerformed(ActionEvent arg0) {

try {

new ViewData(Form.name[0]);

} catch (IOException e) {

}

}

});

}

private static void setLocationCentre(Interface f) {

Dimension us = f.getSize(), them = Toolkit.getDefaultToolkit().getScreenSize();

f.setLocation((them.width - us.width) / 2, (them.height - us.height) / 2);

}

}

class ViewData extends JFrame {

TextArea data;

public ViewData(String nameUser) throws IOException {

super("Log file for User: " + nameUser);

File file = new File(Form.dir + "/" + nameUser + ".log");

BufferedReader buf = new BufferedReader(new FileReader(file));

data = new TextArea();

data.setEditable(false);

add(data, BorderLayout.CENTER);

String line;

while ((line = buf.readLine()) != null) {

data.appendText(line + "\n");

}

buf.close();

this.setSize(400, 400);

this.setLocationCentre(this);

this.setResizable(false);

setVisible(true);

}

private static void setLocationCentre(ViewData f) {

Dimension us = f.getSize(), them = Toolkit.getDefaultToolkit().getScreenSize();

f.setLocation((them.width - us.width) / 2, (them.height - us.height) / 2);

}

}

class TransferData extends Thread {

DateFormat date;

BufferedReader input;

RandomAccessFile acces;

public TransferData(File sourse, File destination) throws IOException {

date = DateFormat.getDateTimeInstance();

input = new BufferedReader(new FileReader(sourse));

acces = new RandomAccessFile(destination, "rw");

acces.seek(destination.length());

String line = null;

acces.writeBytes("[" + date.format(new Date()) + "]:\n");

while ((line = input.readLine()) != null) {

acces.writeBytes(line + "\n");

}

acces.close();

input.close();

start();

}

}

class Form extends Dialog implements ActionListener {

public TextField for\_host = null, for\_name = null;

public Button begin = null;

public String[] host = null;

static public String[] name = null;

Label l\_host = new Label("Host: ");

Label l\_name = new Label("Nume:");

static File dir, file;

public Form(Interface chat, String[] host, String[] name) {

super(chat, "Înregistrare:", true);

setFont(new Font("Comic", Font.BOLD, 12));

setBackground(Color.black);

this.host = host;

this.name = name;

for\_host = new TextField(10);

for\_name = new TextField(10);

Panel for\_texts = new Panel();

for\_texts.add(l\_host);

for\_texts.add(for\_host);

for\_texts.add(l\_name);

for\_texts.add(for\_name);

begin = new Button("Start");

begin.addActionListener(this);

Panel for\_button = new Panel();

for\_button.add(begin);

add(for\_texts, BorderLayout.CENTER);

add(for\_button, BorderLayout.SOUTH);

setSize(180, 125);

setLocationCentre(this);

setResizable(false);

setVisible(true);

}

private static void setLocationCentre(Form f) {

Dimension us = f.getSize(), them = Toolkit.getDefaultToolkit().getScreenSize();

f.setLocation((them.width - us.width) / 2, (them.height - us.height) / 2);

}

@Override

public void actionPerformed(ActionEvent ae) {

host[0] = for\_host.getText();

name[0] = for\_name.getText();

dir = new File("SrvLogs");

if (dir.exists()) {

System.out.println("Directoriul: " + dir.getAbsolutePath() + " exista");

try {

System.out.println("Control la fisier cu logs");

file = new File(dir + "/" + name[0] + ".log");

if (file.exists()) {

System.out.println("Fisierul: " + file.getName() + " exista");

dispose();

} else if (file.createNewFile()) {

System.out.println("Pe server s-a creat fisierul: " + name[0]);

dispose();

}

} catch (IOException e) {

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

}

} else if (dir.mkdir()) {

System.out.println("Directoriul: " + dir.getAbsolutePath() + " s-a creat");

try {

System.out.println("Control la fisier cu logs");

file = new File(dir + "/" + name[0] + ".log");

if (file.exists()) {

System.out.println("Fisierul: " + file.getName() + " exista");

dispose();

} else if (file.createNewFile()) {

System.out.println("Pe server s-a creat fisierul: " + name[0]);

dispose();

}

} catch (IOException e) {

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

}

}

}

}

class Read implements ActionListener {

// Transmiterea mesajului

Interface wnd;

String name = "not :";

Socket ss;

PrintWriter out = null;

String fromUser = "";

DateFormat date;

public Read(Socket s, Interface w) {

wnd = w;

ss = s;

try {

out = new PrintWriter(ss.getOutputStream(), true);

} catch (IOException io) {

}

}

@Override

public void actionPerformed(ActionEvent ae) {

date = DateFormat.getTimeInstance();

fromUser = wnd.to\_send.getText();

wnd.to\_send.setText("");

out.println("[" + date.format(new Date()) + "] " + name + fromUser);

fromUser = "";

}

public void setName(String name) {

this.name = name + " : ";

}

}

public class Client\_Lab4 {

public static void main(String[] args) throws IOException {

String[] name = new String[1], host = new String[1];

Interface wnd = new Interface("Chat");

new Form(wnd, host, name);

Read reader = null;

Socket kkSocket = null;

BufferedReader in = null;

try {

kkSocket = new Socket(host[0], 8888);

in = new BufferedReader(new InputStreamReader(kkSocket.getInputStream()));

} catch (UnknownHostException e) {

System.err.println("Don't know about host.");

System.exit(1);

} catch (IOException e) {

System.err.println("Couldn't get I/O for the connection.");

System.exit(1);

}

String fromServer;

reader = new Read(kkSocket, wnd);

reader.setName(name[0]);

wnd.setTitle(name[0]);

final PrintWriter out = reader.out;

// mesaje de sistem

out.println("^" + name[0]);

wnd.addWindowListener(new WindowAdapter() {

@Override

public void windowClosing(WindowEvent we) {

out.println("2112333");

System.exit(0);

}

});

wnd.b\_send.addActionListener(reader);

wnd.to\_send.addActionListener(reader);

while (((fromServer = in.readLine()) != null) && (reader != null)) {

if ((fromServer.charAt(0) == '^') && (fromServer.charAt(1) == '#')) {

wnd.wnd\_names.setText("");

continue;

}

if (fromServer.charAt(0) == '^') {

wnd.wnd\_names.append(fromServer.substring(1) + '\n');

} else {

wnd.wnd\_msg.append(fromServer + "\n");

}

}

in.close();

kkSocket.close();

}

}



