Client

-ClientPeer

package seriaf.poo.client;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.net.Socket;

import seriaf.poo.structs.Message;

import seriaf.poo.structs.PrivateMessage;

/\*\*

\*

\* @author professor

\*/

public class ClientPeer implements Runnable {

private final ObjectOutputStream mObjectStream;

private final String mSender;

private final Socket socket; // am adaugat socket aici ca sa-l pot folosi mai departe

public ClientPeer(String sender, Socket communicationSocket) throws IOException {

mSender = sender;

mObjectStream = new ObjectOutputStream(communicationSocket.getOutputStream());

this.socket = communicationSocket; // aici trimit socketu sa-l am

}

public void sendMessage(String message) throws IOException {

mObjectStream.writeObject(new Message(mSender, message));

}

public void sendMessage(String recipient, String message) throws IOException {

mObjectStream.writeObject(new PrivateMessage(recipient, mSender, message));

}

public void run() { // metoda pentru thread

try {

ObjectInputStream stream = new ObjectInputStream(this.socket.getInputStream());

while (true) {

System.out.println(stream.readObject().toString().trim());

}

} catch(IOException | ClassNotFoundException e) {

}

}

}

-TextClient

package seriaf.poo.client;

import java.io.IOException;

import java.net.Socket;

import java.util.Scanner;

/\*\*

\*

\* @author professor

\*/

public class TextClient {

private static final String HOST = "127.0.0.1";

private static final int PORT = 9000;

public static void main(String[] args) {

Scanner keyboardScanner = new Scanner(System.in);

try {

Socket communicationSocket = new Socket(HOST, PORT);

String sender = keyboardScanner.nextLine();

ClientPeer peer = new ClientPeer(sender, communicationSocket);

(new Thread(peer)).start(); // am facut un nou thread pt peer caruia i-am dat start

while (true) {

String command = keyboardScanner.nextLine().trim();

if (command.equals("/q")) {

communicationSocket.close();

break;

} else if (command.matches("/w\\s+\\w+\\s+.+")) {

String[] messageParts = command.split("\\s+", 3);

peer.sendMessage(messageParts[1], messageParts[2]);

} else {

peer.sendMessage(command);

}

}

} catch (IOException ex) {

}

}

}

SERVER

->config ,ServerConfig:

package seriaf.poo.server.config;

import seriaf.poo.server.exceptions.MissingKeyException;

import seriaf.poo.server.exceptions.UnknownKeyException;

import seriaf.poo.server.exceptions.InvalidFormatException;

import java.io.FileInputStream;

import java.io.IOException;

import java.util.HashMap;

import java.util.Map;

import java.util.Scanner;

public class ServerConfig {

private Map<String, String> mProperties;

private String[] mKnownProperties = {"TCP\_PORT", "MAX\_CLIENTS"};

public ServerConfig(String filename) throws IOException, InvalidFormatException, UnknownKeyException, MissingKeyException {

mProperties = new HashMap<>();

FileInputStream fileInputStream = new FileInputStream(filename);

Scanner scanner = new Scanner(fileInputStream);

while (scanner.hasNext()) {

String line = scanner.nextLine().trim();

if (line.startsWith("#") || line.isEmpty()) {

continue;

}

if (!line.matches("[a-zA-Z\_][a-zA-Z0-9\_]\*\\s\*=\\s\*[0-9]+")) {

throw new InvalidFormatException("Linia " + line + " nu se potriveste cu formatul asteptat!");

}

processLine(line);

}

for (String property : mKnownProperties) {

if (!mProperties.containsKey(property)) {

throw new MissingKeyException("Cheia " + property + " nu exista in fisier.");

}

}

}

public ServerConfig() throws IOException, InvalidFormatException, UnknownKeyException, MissingKeyException {

this("C:\\Users\\Raishin\\Desktop\\New Folder\\valentin\\src\\seriaf\\poo\\server.conf"); /////////////////////////////

}

private void processLine(String line) throws UnknownKeyException {

String[] words = line.split("=");

String keyName = words[0].trim();

checkKey(keyName);

mProperties.put(keyName, words[1].trim());

}

private void checkKey(String keyName) throws UnknownKeyException {

for (String knownKey : mKnownProperties) {

if (keyName.equals(knownKey)) {

return;

}

}

throw new UnknownKeyException("Cheia " + keyName + " este necunoscuta.");

}

public int getTcpPort() {

return Integer.parseInt(mProperties.get("TCP\_PORT"));

}

public int getMaxClients() {

return Integer.parseInt(mProperties.get("MAX\_CLIENTS"));

}

}

->Exceptions,ex: InvalidFormatException

package seriaf.poo.server.exceptions;

/\*\*

\*

\* @author rhobincu

\*/

public class InvalidFormatException extends Exception {

public InvalidFormatException(String message) {

super(message);

}

}

->Server

package seriaf.poo.server;

import java.io.IOException;

import java.io.ObjectOutputStream;

import java.io.OutputStream;

import java.net.ServerSocket;

import java.net.Socket;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

import java.util.concurrent.CopyOnWriteArrayList;

import seriaf.poo.server.config.ServerConfig;

import seriaf.poo.structs.Message;

import seriaf.poo.structs.PrivateMessage;

/\*\*

\*

\* @author professor

\*/

public class Server {

public ArrayList<ServerPeer> clientList = new ArrayList<>();

private ServerSocket serverSocket;

private final int MAX\_CLIENTS;

public Server(int TCP\_PORT, int MAX\_CLIENTS) { // constructor care instantiaza obiectul serversocket

try {

this.serverSocket = new ServerSocket(TCP\_PORT);

} catch(IOException e) {

}

this.MAX\_CLIENTS = MAX\_CLIENTS;

}

public static void main(String[] args) {

try {

ServerConfig config = new ServerConfig();

Server server = new Server(config.getTcpPort(), config.getMaxClients());

server.listen(server);

} catch(Exception e) {

e.printStackTrace();

}

}

public void listen(Server server) throws IOException {

while(true) { // atata timp cat traieste serveru primeste conexiuni noi

if(clientList.size() <= this.MAX\_CLIENTS) { // atata timp cat mai e loc de conexiuni

ServerPeer peer = new ServerPeer(server, serverSocket.accept());

clientList.add(peer); // tinem minte fiecare client conectat la server ( socketul lui )

(new Thread(peer)).start(); // pornim thread pt serverpeer in loc sa-i dam numai run

}

}

}

public synchronized void dispatch(Message mesaj) {

for(ServerPeer peer : clientList) {

if(mesaj instanceof PrivateMessage) {

if(peer.getUsername().equals(((PrivateMessage) mesaj).getRecipient())) {

try {

ObjectOutputStream objectStream = peer.getObjectStream();

objectStream.writeObject(mesaj);

} catch(Exception e) {

e.printStackTrace();

}

}

} else {

try {

ObjectOutputStream objectStream = peer.getObjectStream();

objectStream.writeObject(mesaj);

} catch(Exception e) {

e.printStackTrace();

}

}

}

}

public synchronized void removeClient(ServerPeer serverPeer) {

clientList.remove(serverPeer);

}

}

->ServerPeer:

package seriaf.poo.server;

import seriaf.poo.structs.Message;

import seriaf.poo.structs.PrivateMessage;

import java.io.EOFException;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.net.Socket;

/\*\*

\*

\* @author professor

\*/

public class ServerPeer implements Runnable {

private final Socket mSocket;

private String username;

private Server server;

private ObjectOutputStream objectStream;

public ServerPeer(Server server, Socket communicationSocket) {

this.mSocket = communicationSocket;

this.server = server;

try {

this.objectStream = new ObjectOutputStream(mSocket.getOutputStream());

} catch(IOException e) {

}

}

public void run() {

Message mesaj;

try {

ObjectInputStream stream = new ObjectInputStream(mSocket.getInputStream());

while (true) {

mesaj = (Message) stream.readObject();

username = mesaj.getSender(); // de fiecare data cand primesti mesaj se updateaza usernameul

System.out.println(mesaj.toString().trim());

if(mesaj instanceof PrivateMessage) {

sendMessage(mesaj);

server.dispatch(mesaj); // trimite mesaju la toata lumea

} else {

server.dispatch(mesaj);

}

}

} catch (EOFException ex) {

// client disconnected gracefully so do nothing

server.removeClient(this); // scoate clientu din lista

} catch (IOException ex) {

server.removeClient(this);

System.err.println("Client connection reset: " + ex.getMessage());

} catch (ClassNotFoundException ex) {

System.err.println("Unknown object received.");

}

}

private void sendMessage(Message mesaj) {

try {

objectStream.writeObject(mesaj); // serializeaza mesajul si il trimite catre socket

} catch(IOException e) {

}

}

public Socket getmSocket() {

return this.mSocket;

}

public String getUsername() { //this is pretty obv as well

return this.username;

}

public ObjectOutputStream getObjectStream() {

return this.objectStream;

}

}

STRUCTS:

->Main

package seriaf.poo.structs;

import java.io.IOException;

import java.util.logging.Level;

import java.util.logging.Logger;

import seriaf.poo.server.config.ServerConfig;

import seriaf.poo.server.exceptions.InvalidFormatException;

import seriaf.poo.server.exceptions.MissingKeyException;

import seriaf.poo.server.exceptions.UnknownKeyException;

/\*\*

\*

\* @author professor

\*/

public class Main {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

Message m1 = new Message("John", "Hi!");

Message m2 = new Message("Mary", "Hello!");

Message m3 = new Message("John", "Bye!");

Message m4 = new Message("Mary", "Awww...");

System.out.println(m1.toString());

System.out.println(m2);

System.out.printf("%s\n", m3);

System.out.println(m4);

PrivateMessage pm1 = new PrivateMessage("Mary", "John", "Hi!");

PrivateMessage pm2 = new PrivateMessage("John", "Mary", "Leave me alone, I have a restraining order!");

PrivateMessage pm3 = new PrivateMessage("Mary", "John", "Aww, come on, don't be like that!");

System.out.println(pm1);

System.out.println("To: " + pm1.getRecipient());

System.out.println(pm2);

System.out.println("To: " + pm2.getRecipient());

System.out.println(pm3);

System.out.println("To: " + pm3.getRecipient());

try {

ServerConfig config = new ServerConfig();

System.out.println("Tcp port: " + config.getTcpPort());

System.out.println("Max clients: " + config.getMaxClients());

} catch (IOException ex) {

System.err.println("Exceptie de tip IO: " + ex.getMessage());

} catch (InvalidFormatException ex) {

System.err.println("Format invalid: " + ex.getMessage());

} catch (UnknownKeyException | MissingKeyException ex) {

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

}

}

}

->Message

package seriaf.poo.structs;

import java.io.Serializable;

/\*\*

\*

\* @author professor

\*/

public class Message implements Serializable{

private static final long serialVersionUID = 1L;

private String mSender;

private String mContent;

public Message(String sender, String content) {

mSender = sender;

mContent = content;

}

@Override

public String toString() {

return mSender + ": " + mContent;

}

public String getSender() { // returneaza senderul.... pretty obvious

return this.mSender;

}

}

->Private Message

package seriaf.poo.structs;

/\*\*

\*

\* @author rhobincu

\*/

public class PrivateMessage extends Message {

private static final long serialVersionUID = 1L;

private final String mRecipient;

public PrivateMessage(String recipient, String sender, String content) {

super(sender, content);

mRecipient = recipient;

}

@Override

public String toString() {

return "(priv) " + super.toString();

}

public String getRecipient() {

return mRecipient;

}

}