Tema Programare Orientata pe Obiecte

Student: Duminica Alex Vasile
Anul: 3
Facultatea: Calculatoare
Grupa: 1

Tema

Descriere a Funcționalității Aplicației

Aceste două aplicații Java reprezintă un simplu exemplu de comunicare client-server. Aplicația client trimite o oră în formatul "hh:mm:ss" la server, iar serverul convertește acea oră în secunde și îi trimite înapoi clientului rezultatul.

Aplicația 1 - Client:

1. Se creează un obiect `Scanner` pentru a citi inputul de la utilizator.
2. Se creează un obiect `Socket` pentru a se conecta la server la adresa "localhost" și portul 5000.
3. Se deschid fluxurile de citire (`BufferedReader`) și scriere (`PrintWriter`) pentru comunicarea cu serverul.
4. Se afișează mesaje de conectare și instrucțiuni pentru utilizator.
5. Se intră într-un buclu care primește input de la utilizator, trimite serverului și așteaptă răspunsul.
6. Bucla rulează până când utilizatorul introduce "STOP".
7. După încheierea buclei, se închid toate resursele și se afișează un mesaj de deconectare.

Aplicația 2 - Server:

1. Se creează un obiect `ServerSocket` care ascultă pe portul 5000.
2. Se așteaptă conexiuni de la clienți într-un buclu infinit.
3. Odată ce un client se conectează, se creează un obiect `HandleClient` (care este o clasă separată) și se pornește un fir de execuție (`Thread`) pentru a gestiona comunicarea cu clientul.
4. Se afișează informații despre clientul conectat.
5. Bucla continuă să aștepte noi conexiuni.

Aplicația 2 - Clasa HandleClient:

1. Constructorul primește un socket de la client și inițializează fluxurile de citire și scriere.
2. Metoda `run` este apelată când firul de execuție începe să ruleze. Aici se primește input de la client, se procesează, și se trimite înapoi rezultatul.
3. Dacă clientul introduce "STOP", bucla se încheie, și resursele sunt închise.
4. Se verifică validitatea inputului și se tratează cazurile de eroare.
5. Se închid resursele (socket, fluxuri de citire și scriere) când comunicarea cu clientul este finalizată.

Client (client/Client.java)

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.util.Scanner;

public class Client {
    public static void main(String[] args) throws IOException {
        Scanner scanner = new Scanner(System.in);
        Socket clientSocket = null;
        BufferedReader bufferedReader = null;
        PrintWriter printWriter = null;

        try {
            clientSocket = new Socket("localhost", 5000);
                                InputStreamReader  inputStreamReader  =  new
InputStreamReader(clientSocket.getInputStream());
            bufferedReader = new BufferedReader(inputStreamReader);
            printWriter = new PrintWriter(clientSocket.getOutputStream());

            System.out.println("Connected!");

            System.out.println("Type a time in format hh:mm:ss");
             System.out.println("The server will send back the time converted
in seconds.");

            while (true) {
                System.out.print("Time: ");
                String timeString = scanner.nextLine();

                if (timeString.contains("STOP"))
                    break;

                printWriter.println(timeString);
                printWriter.flush();

                String responseString = bufferedReader.readLine();
                System.out.println(responseString);
            }

            clientSocket.close();
            scanner.close();
        } catch (IOException e) {
            e.printStackTrace();
        }

        System.out.println("Disconnected!");
    }
}
```

Server (server/Server.java)

```
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import HandleClient.HandleClient;

public class Server {

    public static void main(String[] args) {

        try (ServerSocket serverSocket = new ServerSocket(5000)) {
            System.out.println("Server Online!");

            while (true) {
                Socket clientSocket = serverSocket.accept();
                HandleClient handleClient = new HandleClient(clientSocket);
                handleClient.start();
                    System.out.println(clientSocket.getLocalAddress() + ":" +
clientSocket.getLocalPort() + " connected!");
            }
        } catch (IOException e) {
            e.printStackTrace();
        }

    }

}
```

HandleClient (server/HandleClient/HandleClient.java)

```
package HandleClient;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;

public class HandleClient extends Thread {
    private Socket clientSocket;
    private BufferedReader bufferedReader;
    private PrintWriter printWriter;

    public HandleClient(Socket clientSocket) {
        try {
            this.clientSocket = clientSocket;
            printWriter = new PrintWriter(clientSocket.getOutputStream());
                                InputStreamReader inputStreamReader = new
InputStreamReader(clientSocket.getInputStream());
            this.bufferedReader = new BufferedReader(inputStreamReader);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

```
    public void run() {
        try {
            String clientInput;
            while ((clientInput = bufferedReader.readLine()) != null) {
                if (clientInput.equals("STOP")) {
                    break;
                }

                String[] data = clientInput.split(":");
                if (data.length != 3) {
                    // Handle invalid input
                        String errorMessage = "Error: Time format should be
hh:mm:ss!";

                    System.err.println(errorMessage);

                    printWriter.println(errorMessage);
                    printWriter.flush();

                    continue;
                }

                try {
                    int hours = Integer.parseInt(data[0]);
                    int minutes = Integer.parseInt(data[1]);
                    int seconds = Integer.parseInt(data[2]);

                        int timeInSeconds = hours * 3600 + minutes * 60 +
seconds;

                    printWriter.println(Integer.toString(timeInSeconds));
                    printWriter.flush();
                } catch (NumberFormatException e) {
                    // Handle invalid input
                      String errorMessage = "Invalid numeric input: The input
should contain only numeric values for hours, minutes, and seconds in the
format hh:mm:ss!";

                    System.err.println(errorMessage);

                    printWriter.println(errorMessage);
                    printWriter.flush();
                }
            }
        } catch (IOException e) {
            // Handle IOException
            e.printStackTrace();
        } finally {
            // Close resources if needed
            try {
                    System.out.println(clientSocket.getLocalAddress() + ":" +
clientSocket.getLocalPort() + " disconnected!");
                bufferedReader.close();
```

```
                printWriter.close();
                clientSocket.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

    }
}
```