Proiect Programare Orientata pe Obiecte

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

Aplicatia 2

Descriere a Funcționalității Aplicației 2

Aceste două aplicații Java creează un simplu sistem de gestionare a cărților cu un server și un client care pot comunica între ele prin rețea. Sistemul este conceput pentru a gestiona stocul de cărți într-o bază de date și pentru a permite clienților să solicite informații despre disponibilitatea cărților.

Aplicația 1 - Client:

- Creează un socket pentru a se conecta la serverul care rulează pe localhost (adresa IP a mașinii locale) și portul 5000.
- Utilizează `BufferedReader` pentru a citi de la server și `PrintWriter` pentru a scrie la server.
- Prin intermediul unui ciclu `while`, primește input de la utilizator referitor la autor și titlu, trimite aceste informații la server și afișează răspunsul primit de la server.
- Ciclul continuă până când utilizatorul introduce "STOP".

Aplicația 2 - Server:

- Așteaptă conexiuni de la clienți pe portul 5000.
- La fiecare conexiune acceptată, lansează un nou thread (`HandleClient`) pentru a manipula comunicația cu clientul respectiv.
- `HandleClient` primește informații de la client, caută în baza de date și returnează stocul cărții solicitate.
- Folosește o clasă `Database` pentru a încărca datele despre cărți dintr-un fișier text și pentru a oferi funcționalități de căutare.
- La fiecare conexiune, afișează adresa IP și portul clientului și continuă să aștepte alte conexiuni.

Pachetul HandleClient:

- `HandleClient` este un fir de execuție separat care gestionează comunicarea cu un client.
- Primește datele de la client, le procesează în baza de date și trimite înapoi stocul cărții.

Pachetul Database:

- `Database` încarcă inițial datele dintr-un fișier text și oferă funcționalități pentru căutarea cărților în baza de date.
- `Entry` reprezintă o înregistrare individuală pentru o carte, cu informații precum autor, titlu și stoc.

Fișierul books.txt:

- Conține datele inițiale ale cărților, precum autorul, titlul și stocul.

În esență, aplicațiile implementează un sistem simplu de gestiune a stocului de cărți într-o bază de date și permit clienților să solicite informații despre disponibilitatea cărților prin intermediul rețelei. Aplicația server utilizează fire de execuție pentru a gestiona mai mulți clienți simultan.

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 an author name and book title");
            System.out.println("The server will send back the book stock.");

            while (true) {
                System.out.print("Author: ");
                String authorString = scanner.nextLine();
                System.out.print("Title: ");
                String titleString = scanner.nextLine();

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

                printWriter.println(authorString + ":" + titleString);
                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 Database.*;
import HandleClient.HandleClient;

public class Server {

    public static void main(String[] args) {

        try (ServerSocket serverSocket = new ServerSocket(5000)) {
            Database database = new Database("books.txt");
            System.out.println("Server Online!");
            database.printEntries();

            while (true) {
                Socket clientSocket = serverSocket.accept();
                   HandleClient handleClient = new HandleClient(clientSocket,
database);
                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;

import Database.*;

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

    public HandleClient(Socket clientSocket, Database database) {
        try {
            this.database = database;
            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(":");
                Entry entry = database.getEntry(data[0], data[1]);

                printWriter.println(entry.getStock().toString());
                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();
            }
        }

    }
}
```

Server Database

Database (server/Database/Database.java)

```
package Database;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;

public class Database {
    String FilePath;
    ArrayList<Entry> Entries;

    public Database(String Path) {
        this.FilePath = Path;
        this.Entries = new ArrayList<Entry>();
        loadDatabase();

    }

    void loadDatabase() {
                    try (BufferedReader reader = new BufferedReader(new
FileReader(this.FilePath))) {
            String line;

            while ((line = reader.readLine()) != null) {
                String title = line.trim();
                String author = reader.readLine().trim();
                int stock = Integer.parseInt(reader.readLine().trim());

                Entry entry = new Entry(author, title, stock);
                Entries.add(entry);
            }
        } catch (IOException | NumberFormatException e) {
            e.printStackTrace();
        }
    }

    public Entry getEntry(String Author, String Title) {
        Entry search = new Entry(Author, Title, null);
        for(Entry entry : Entries) {
            if(entry.equals(search))
                return entry;
        }

        return search;
    }
```

```
    public void printEntries(){
        for(Entry entry : this.Entries) {
            System.out.println(entry.getAuthor() + ":" + entry.getTitle() +
":" + entry.getStock());
        }
    }
}
```

Entry (server/Database/Entry.java)

```
package Database;

public class Entry {
    String Author;
    String Title;
    Integer Stock;

    public Entry(String Author, String Title, Integer Stock) {
        this.Author = Author;
        this.Title = Title;
        this.Stock = Stock;
    }

    public String getAuthor() {
        return this.Author;
    }

    public String getTitle() {
        return this.Title;
    }

    public Integer getStock() {
        return this.Stock;
    }

    public boolean equals(Entry entry) {
                    return  entry.getAuthor().contentEquals(this.Author)     &&
entry.getTitle().contentEquals(this.Title);
    }
}
```

books.txt (server/books.txt)

```
Amintiri din copilarie
Ion Creanga
10
Marile sperante
Charles Dickens
7
```