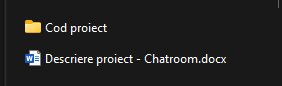
Java Sockets Chatroom

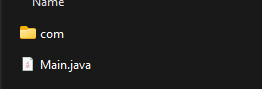
* Bărdiță Andrei

Pentru a gestiona mai mulți clienți simultan într-o aplicație de chat, putem utiliza un model de server cu fire de execuție multiple (multi-threaded). Fiecare client care se conectează la server va primi un fir de execuție dedicat pentru a citi mesajele de la client și pentru a le transmite celorlalți clienți conectați.

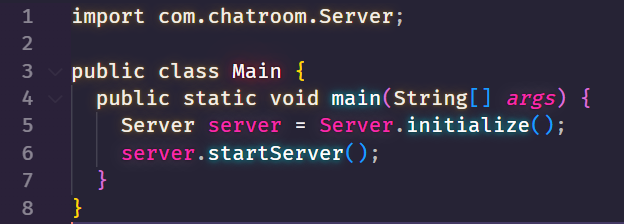
În locația curentă există directorul “Cod proiect” care conține codul sursă al aplicației.



Pentru a rula această aplicație este nevoie de cel puțin 2 instanțe deschise, una pentru server, iar cealaltă pentru client. Serverul se va deschide rulând fișierul *Main.java*, iar clientul cu fișierul *Client.java* din com.chatroom.



Clasa ***Main*** instanțiază și pornește serverul, o clasă Singleton.



De acum încolo ne vom muta în directorul com.chatroom.

**Clasa Server**



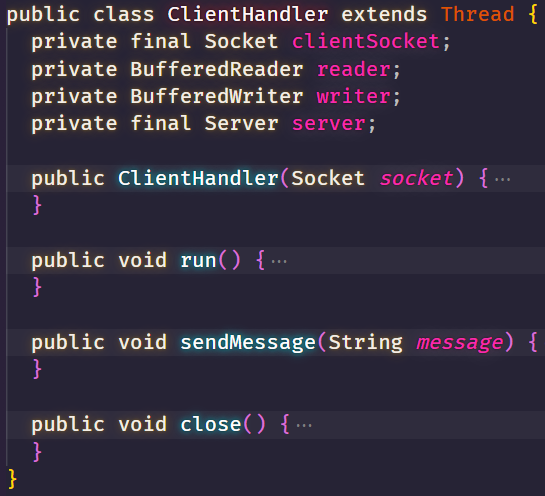
Primele două metode din clasa ***Server*** sunt pentru inițializare. După ce serverul a fost inițializat în clasa ***Main***, acesta va trebui să fie pornit cu metoda *startServer*. Aceasta creează lista de clienți, stabilește portul și așteaptă conexiuni de la clienți folosind obiectul "ServerSocket". Odată ce un client se conectează, se afișează adresa IP a clientului și se creează un obiect "***ClientHandler***" pentru a trata comunicarea cu acel client. Obiectul "***ClientHandler***" este adăugat la lista de clienți și pornit într-un fir de execuție separat.

Metoda "*closeServer*" este responsabilă cu închiderea serverului. Aceasta închide toți clienții conectați prin apelul metodei "*close*" pe fiecare obiect "***ClientHandler***" și apoi închide socketul serverului.

Metoda "*removeClient*" este folosită pentru a elimina un client din lista de clienți atunci când acesta se deconectează.

Metoda "*broadcastMessage*" este responsabilă pentru transmiterea unui mesaj către toți clienții conectați. Mesajul este trimis fiecărui client prin metoda "*sendMessage*".

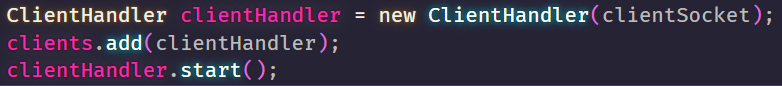
**Clasa ClientHandler**

****

Am creat clasa ***ClientHandler***, care extinde clasa ***Thread*** (un fir de execuție separat pentru fiecare client conectat la serverul de chat) și gestionează comunicarea cu un client specific. Fiecare client nou care se conectează la server primește un obiect ***ClientHandler*** și este adăugat în lista de clienți conectați.

Constructorul clasei primește un obiect "Socket" ca argument, care reprezintă conexiunea cu clientul. În constructor, se inițializează socketul clientului și se obține o referință la instanța serverului prin apelul metodei "*initialize*" a clasei "***Server***".

Metoda "*run*" reprezintă punctul de intrare în firul de execuție al handler-ului de client și pornește odată cu apelarea metodei start din clasa ***Server***.

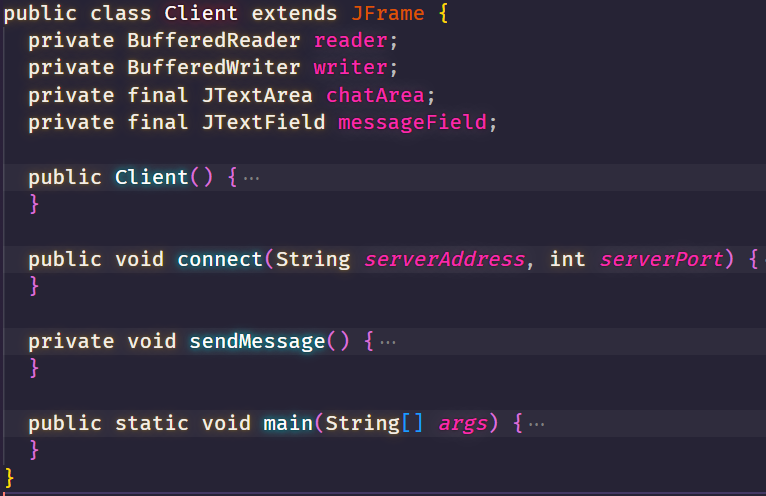


Aici se realizează citirea mesajelor de la client și transmiterea lor către server pentru a fi difuzate către toți clienții conectați. Firul de execuție rulează într-o buclă infinită până când clientul închide conexiunea sau apare o excepție de tip **IOException**. În fiecare iterație a buclei, se citește un mesaj de la client, se afișează în consolă și se transmite către server pentru difuzare. În cazul în care apare o excepție de tip **IOException** în timpul comunicării cu clientul, aceasta este afișată. În final, indiferent dacă firul de execuție se încheie normal sau datorită unei excepții, se apelează metoda "*close*" pentru a închide resursele asociate cu citirea, scrierea și socketul clientului.

Metoda "*sendMessage*" este folosită pentru a trimite un mesaj către client. Mesajul este scris în buffer-ul de ieșire, apoi este trimis către client prin apelurile "*newLine*" și "*flush*".

Într-un final, metoda "*close*" închide resursele asociate citirii, scrierii și socketului clientului. În primul rând, se închide obiectul "reader" și "writer". Apoi, se închide socketul clientului și se apelează metoda "*removeClient*" a serverului pentru a elimina handler-ul de client din lista de clienți conectați la server.

**Clasa Client**



Această clasă ***Client*** extinde ***JFrame*** și conține o interfață grafică simplă cu o zonă de chat (reprezentată de un ***JTextArea***) și un câmp de mesaje (reprezentat de un ***JTextField***). Când utilizatorul apasă Enter în câmpul de mesaje sau face clic pe butonul "Send", mesajul este trimis la server prin apelul metodei "*sendMessage*", iar mesajele primite de la server sunt afișate în zona de chat.

Constructorul clasei inițializează fereastra, setând titlul, dimensiunea și comportamentul de închidere. Se crează un panou pentru afișarea istoricului de chat, adăugând un obiect ***JTextArea*** într-un ***JScrollPane***. Panoul este adăugat în partea centrală a ferestrei. Se crează un panou pentru introducerea mesajului, adăugând un obiect ***JTextField*** și un buton "*Send*". Butonul și câmpul de text au asociați ascultători de evenimente care răspund atunci când utilizatorul apasă Enter sau butonul "*Send*". Panoul este adăugat în partea de jos a ferestrei. Se adaugă panourile la fereastra principală folosind un layout de tip ***BorderLayout***.

Metoda "*connect*" este utilizată pentru a realiza conexiunea cu serverul de chat. Prin furnizarea adresei IP a serverului și portului, se creează un obiect "Socket" și se inițializează readerul și writerul pentru comunicarea cu serverul. De asemenea, se creează un fir de execuție separat pentru a citi mesajele primite de la server și a le afișa în zona de chat.

Metoda "*sendMessage*" este utilizată pentru a trimite mesajul introdus de utilizator către server. Mesajul este citit din câmpul de text, iar dacă nu este gol, este scris în buffer-ul de ieșire și trimis către server. După trimitere, câmpul de text este golit.

Metoda "*main*" reprezintă punctul de intrare în aplicație. Se creează o instanță a clasei "***Client***" și se apelează metoda "*connect*" pentru a stabili conexiunea cu serverul. Această secțiune de cod rulează pe firul de execuție al interfeței grafice Swing utilizând metoda "*invokeLater*". În metoda "*main*", am folosit "*invokeLater*" pentru a asigura că aplicația GUI este inițializată și afișată pe firul de execuție GUI principal.

**Concluzie**

Am dezvoltat aplicația de chatroom, utilizând o clasă Main care deschide Serverul și 3 clase în package-ul com.chatroom pentru a împărți pe atribute fiecare funcționalitate.

Server.java și ClientHandler.java țin de partea de server, iar Client.java de cea de client.

Se pot deschide oricâte instanțe de client se dorește cu ajutorul clasei Client, dar poate exista doar o singură instanță pentru clasa Server (datorită tiparului Singleton).