Documentation du Chat UDP

Multi-Clients

# 1. Introduction

Ce projet implémente un chat multi-clients en UDP où un serveur fait transiter les messages entre les clients. Chaque client peut envoyer un message à un autre en suivant le format :

destinataire:message

Les clients peuvent également se déconnecter proprement en envoyant "exit", et le serveur les supprime de sa liste.

# 2. Structure du Projet

Fichiers principaux :

1. UDPServer.java – Gère les connexions, les messages et la déconnexion des clients.

2. UDPClient.java – Permet aux utilisateurs de se connecter et d’échanger des messages.

# 3. Fonctionnement

### 3.1. Lancement du Serveur

Le serveur écoute sur le port 5432 et maintient une liste des clients connectés à l’aide d’une HashMap. Il reçoit des messages et les redirige ensuite aux destinataires.

### 3.2. Connexion d’un Client

Chaque client doit entrer un nom d’utilisateur à la connexion.

Le serveur enregistre ce nom et l’adresse du client.

### 3.3. Échange de Messages

Un client envoie un message sous la forme destinataire:message.

Le serveur redirige le message au destinataire.

Si le destinataire n’existe pas, le serveur informe l’émetteur.

### 3.4. Déconnexion

Si un client envoie "exit", le serveur le supprime de la liste des clients connectés.

Un message est affiché dans la console du serveur indiquant la déconnexion.

# 4. Détails des Classes et Fonctions

### 4.1. Serveur (UDPServer.java)

Gère la communication entre les clients.

Attributs :

* PORT : Port d’écoute du serveur (5432).
* clients : HashMap qui stocke les utilisateurs connectés (nom -> InetSocketAddress).

Méthodes :

* main(String[] args) : Démarre le serveur et attend les messages des clients.
* handleMessage(DatagramSocket socket, String message, InetSocketAddress clientAddress) : Gère les messages reçus (connexion, envoi de message, déconnexion)
* sendMessage(DatagramSocket socket, String message, InetSocketAddress address) : Envoie un message à un client donné par l’utilisateur.
* removeClient(InetSocketAddress address) : Supprime un client de la liste en cas de déconnexion

### 4.2. Client (UDPClient.java)

Chaque utilisateur exécute un client qui permet de se connecter au serveur et d’échanger des messages. Il peut y avoir plusieurs clients en simultané.

Méthodes :

* main(String[] args) : Démarre le client, demande un nom d'utilisateur, envoie et reçoit des messages.
* sendMessage(DatagramSocket socket, InetAddress address, String message) : Envoie un message au serveur.
* receiveMessages (DatagramSocket socket) : Écoute les messages reçus du serveur en arrière-plan.

# 5. Exemple d’Utilisation

### 5.1. Démarrer le Serveur

Compiler et exécuter :

* javac UDPServer.java
* java UDPServer

### 5.2. Démarrer Plusieurs Clients

Compiler et exécuter :

* javac UDPClient.java
* java UDPClient

### 5.3. Envoi de Messages

Message (format: destinataire:message ou 'exit' pour quitter) : Quentin:Salut

### 5.4. Déconnexion d’un Client

Message (format: destinataire:message ou 'exit' pour quitter) : exit