AILLAUD Romain
DELAVAUD Quentin

Documentation du Chat UDP Multi-Clients

1. Introduction

Ce projet met en place un chat multi-clients en UDP, où un serveur central gère les connexions et permet aux clients d'échanger des messages entre eux.

Les clients peuvent :

- Envoyer un message à un autre utilisateur (destinataire:message)
- Envoyer un message en broadcast à tout le monde (*:message)
- Demander la liste des utilisateurs connectés (list)
- Se déconnecter proprement (exit)

Le serveur gère automatiquement les connexions et déconnexions.

2. Structure du Projet

Fichiers principaux :

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

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 envoyer son nom précédé d'un + pour s'enregistrer (+romain).

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

3.3. Échange de Messages

- **Message privé** : destinataire:message

Le serveur redirige le message au destinataire.

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

- **Message à tout le monde (broadcast)** : *:message
- **Demande de la liste des clients** : list

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)

Le serveur écoute les messages des clients et les redirige aux destinataires. Il gère aussi les commandes spéciales.

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, gère les commandes (list, exit, +nom).
- **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 Connexion d'un Client

Envoyer :

- +romain

Le serveur répond :

- Bienvenue romain! Utilisez format: destinataire:message

5.4. Envoi de Messages

A une personne spécifique :

- romain:Salut Romain

A tout le monde (broadcast) :

- *:Salut tout le monde

5.5. Liste des Clients Connectés

Envoyer :

- list

Le serveur répond :

- [romain, quentin]

5.6. Déconnexion d’un Client

Envoyer :

- exit