

AI16 Projet

Application de chat multi-thread en Java

Lien vers le repo:

Le projet est disponible en suivant le lien ci-contre :

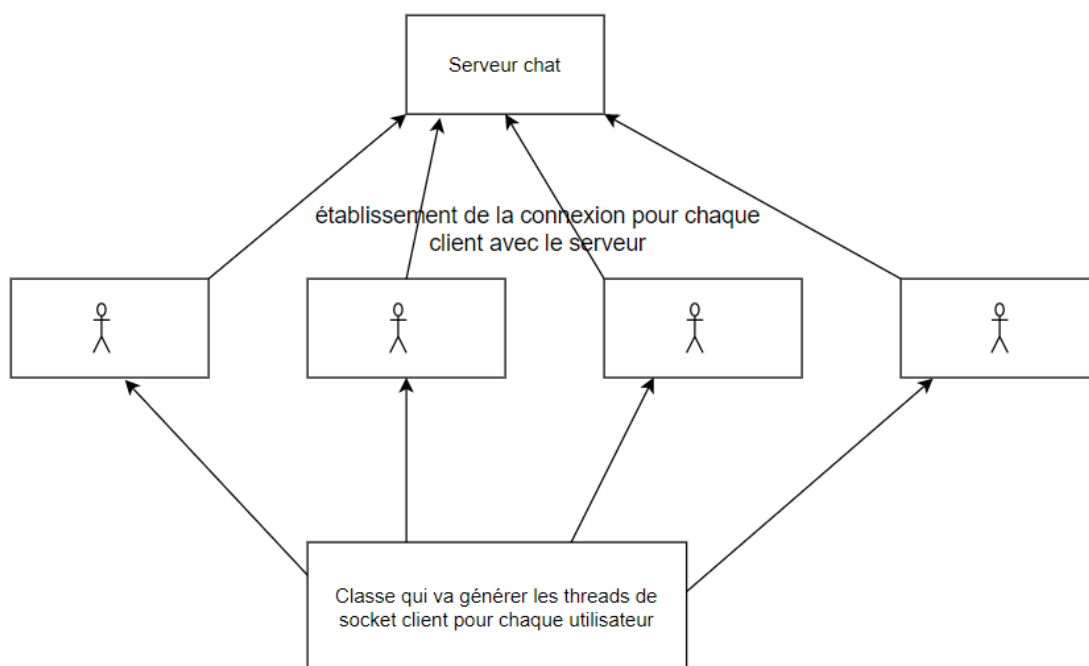
https://gitlab.utc.fr/elimouni/ai16-douale-grelrier-limouni/-/tree/master/projet_tchat_socket

Vous trouverez, en haut des fichiers JAVA un commentaire récapitulant les différentes répartitions du travail au sein du groupe.

Contexte :

Le projet consiste en la création d'une application multi-thread permettant à plusieurs personnes de communiquer sur un chat tous ensemble. Ce projet met en avant le principe de multi-threading, qui permet d'exécuter plusieurs tâches en parallèle. Ici, le multi-threading permet de gérer les sockets des différents utilisateurs qui vont se connecter au chat. Le principe de socket est donc aussi mis en avant, et un socket est un objet java qui permet de gérer une communication entre un client et un serveur. Enfin, on utilisera une hashmap, qui est une structure de données, qui va nous permettre de stocker les informations sur chaque socket créée par thread, afin de pouvoir faire en sorte que les messages envoyés par les clients soient envoyés à toutes les personnes connectées.

Voici un schéma théorique explicatif du fonctionnement de notre application de multi-threading :



Pour chaque utilisateur qui va se connecter au chat, il va récupérer les informations sur les sockets des autres clients déjà créés, qui se trouvent dans une hashmap.

Notre choix de conception et de développement :

Nous avons créé les classes suivantes:

- ClientChat
- ClientChat2
- ClientReceptor
- ServerChat
- MessageReceptor

Les deux classes *ClientChat* et *ClientChat2* permettent d'instancier deux clients différents et faire des tests sur la communication multi threads. Ces derniers demandent une connexion au serveur, puis peuvent communiquer avec ce dernier. Afin de pouvoir afficher les messages diffusés par les autres utilisateurs, une classe *ClientReceptor* permet de paralléliser (en étendant la classe Thread) la réception des messages et l'envoi, permettant d'obtenir dans un seul et même terminal la suite des messages envoyés tout en permettant l'envoi.

La classe *ServerChat* permet de créer un serveur central gérant les clients. Il s'agit d'un point d'arrivée de tous les clients. Pour permettre la gestion de plusieurs clients, un thread par client est ouvert, grâce à la classe *MessageReceptor*, rendant indépendante la gestion de chaque client.

La classe *MessageReceptor* permet la gestion des messages entrants de la part des clients, du formatage à la diffusion à tous les autres clients connectés.

Pour stocker les adresses dans *ServerChat*, nous avons opté pour une `HashMap<Socket, String>`, qui permet d'assigner à un socket un pseudonyme utilisateur. Cette structure assure l'unicité des sockets clients stockés, et a une taille qui est dynamiquement augmentable.

Afin de pouvoir utiliser les Threads, nous avons utilisé `@Override` sur les méthodes `run()`, ce qui a permis de réécrire ces fonctions suite à l'extension de la classe Thread.

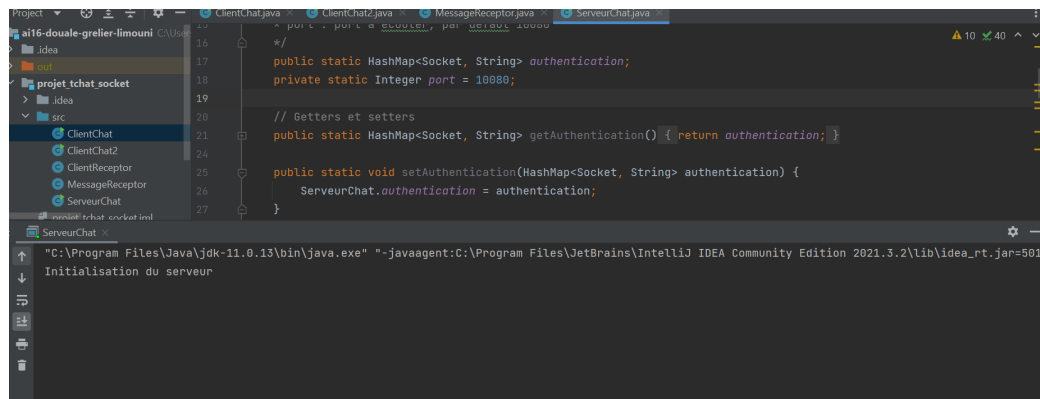
Scénarios :

Le code source est récupérable en suivant le lien Git présenté dans la première partie. Il suffira alors de le cloner et de le lancer.

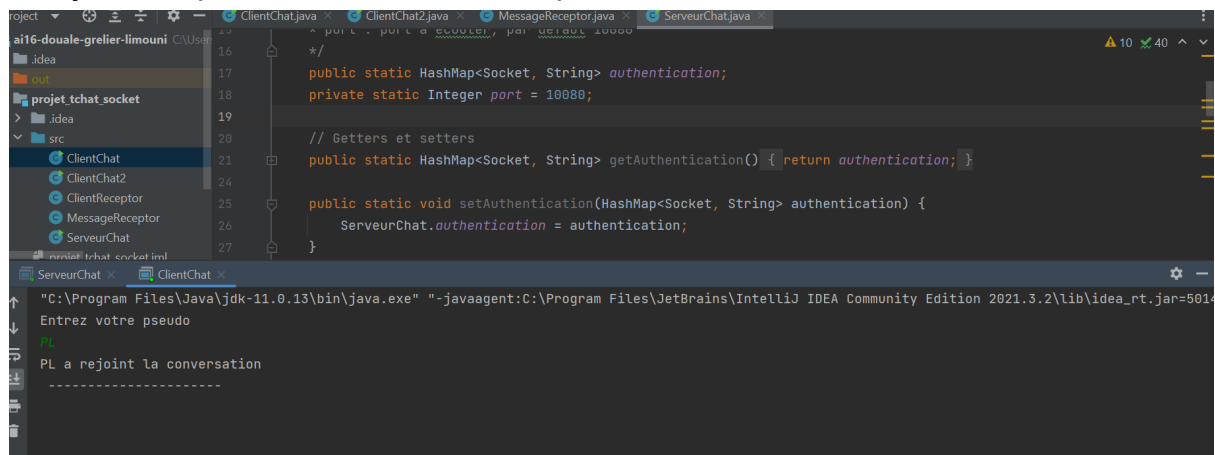
Pour utiliser le programme, il suffit de lancer la classe *ServerChat*, suivie d'autant d'instanciations de *ClientChat* que voulues. Une seconde classe *ClientChat2* est disponible à titre de démo pour créer deux clients depuis un même IDE.

Voici une démo de la communication entre deux clients:

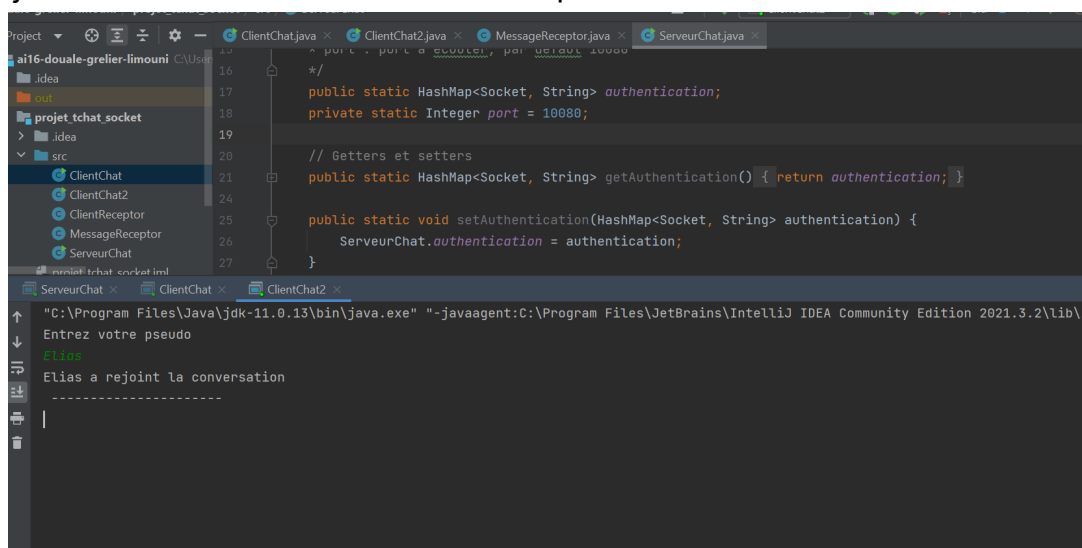
On lance le serveur:



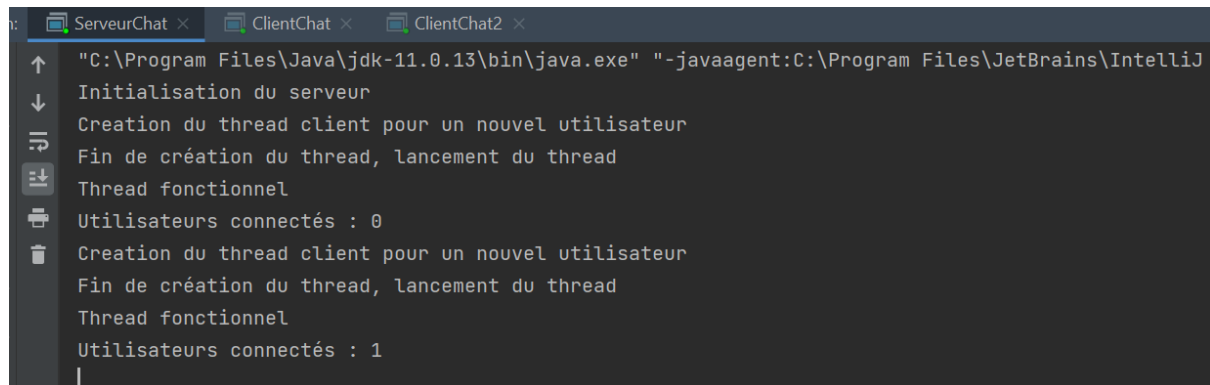
On ajoute un premier utilisateur "PL" à partir de ClientChat:



On ajoute un deuxième utilisateur "Elias" à partir de ClientChat2:

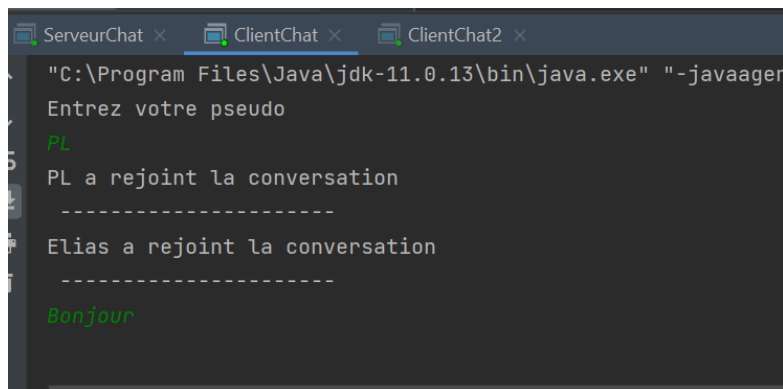


On observe du côté serveur les deux connexions.



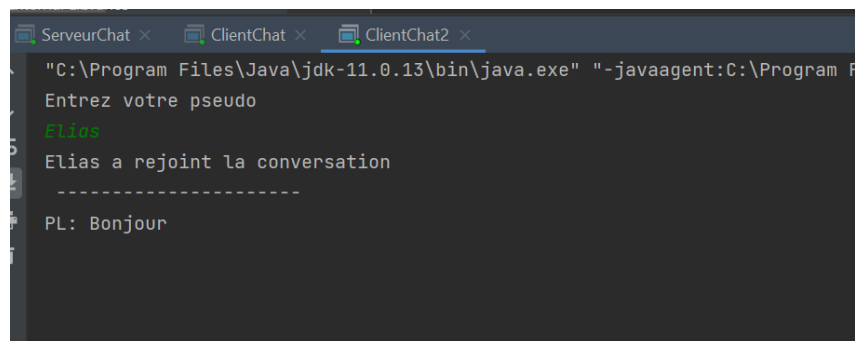
```
"C:\Program Files\Java\jdk-11.0.13\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ
Initialisation du serveur
Creation du thread client pour un nouvel utilisateur
Fin de création du thread, lancement du thread
Thread fonctionnel
Utilisateurs connectés : 0
Creation du thread client pour un nouvel utilisateur
Fin de création du thread, lancement du thread
Thread fonctionnel
Utilisateurs connectés : 1
```

“PL” écrit un message:



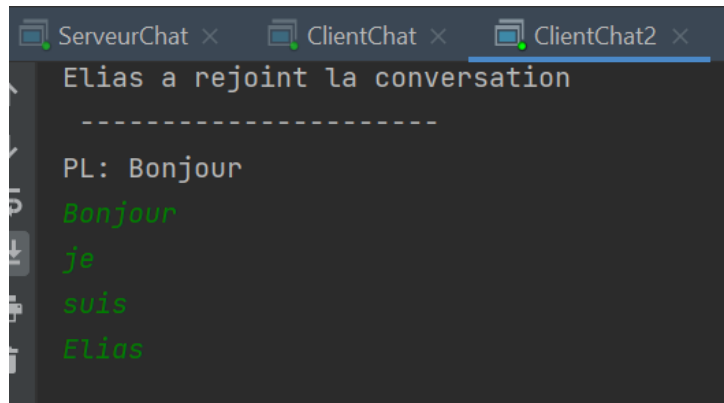
```
"C:\Program Files\Java\jdk-11.0.13\bin\java.exe" "-javaagen
Entrez votre pseudo
PL
PL a rejoint la conversation
-----
Elias a rejoint la conversation
-----
Bonjour
```

Sur le chat de “Elias” on voit apparaître le message de PL:



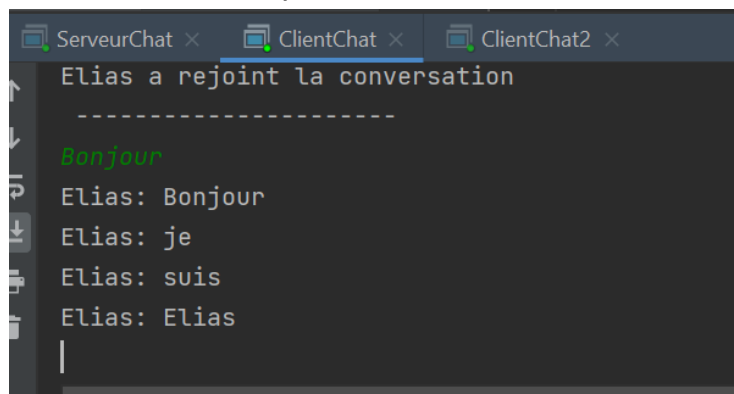
```
"C:\Program Files\Java\jdk-11.0.13\bin\java.exe" "-javaagent:C:\Program F
Entrez votre pseudo
Elias
Elias a rejoint la conversation
-----
PL: Bonjour
```

“Elias” répond à PL:



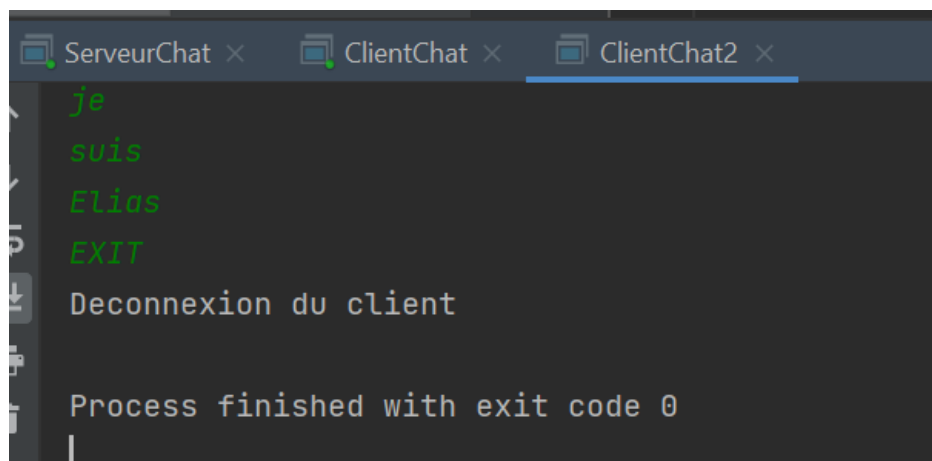
```
ServeurChat x ClientChat x ClientChat2 x
Elias a rejoint la conversation
-----
PL: Bonjour
Bonjour
je
suis
Elias
```

On observe sur le chat de “PL” la réponse de “Elias”



```
ServeurChat x ClientChat x ClientChat2 x
Elias a rejoint la conversation
-----
Bonjour
Elias: Bonjour
Elias: je
Elias: suis
Elias: Elias
|
```

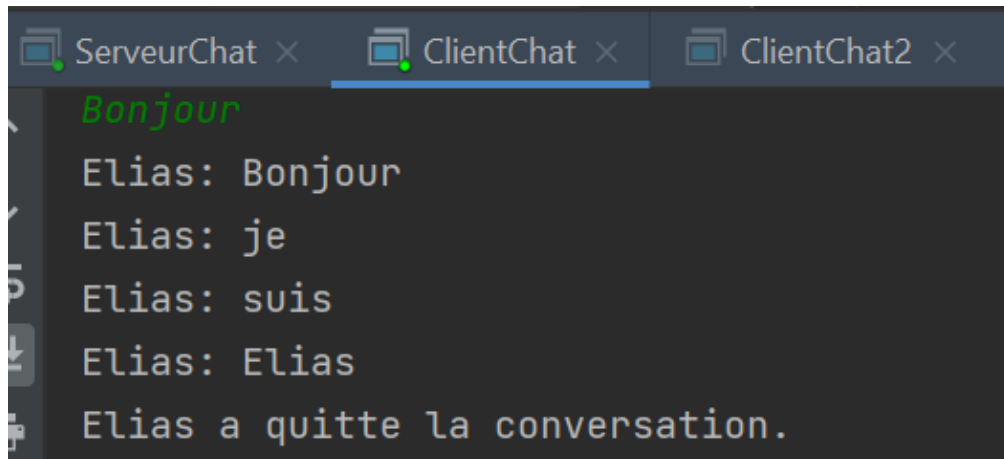
Déconnexion de “Elias” avec la commande EXIT:



```
ServeurChat x ClientChat x ClientChat2 x
je
suis
Elias
EXIT
Deconnexion du client

Process finished with exit code 0
|
```

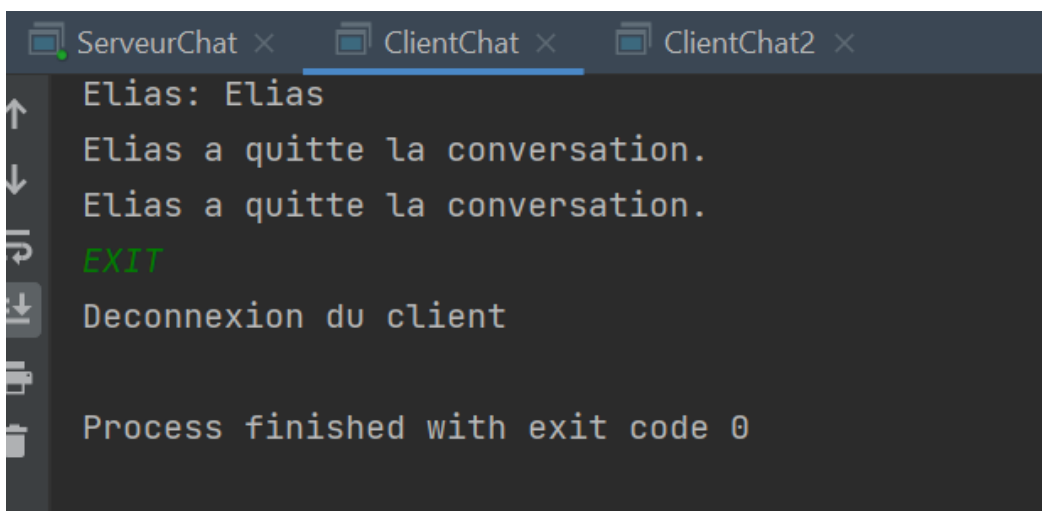
On observe sur le chat de “PL” la déconnexion de “Elias”:



The screenshot shows a window with three tabs: 'ServeurChat', 'ClientChat', and 'ClientChat2'. The 'ClientChat' tab is active. The chat history shows the following messages:

```
Bonjour
Elias: Bonjour
Elias: je
Elias: suis
Elias: Elias
Elias a quitte la conversation.
```

Déconnexion de "PL":



The screenshot shows the same window as before, but the 'ClientChat' tab now displays the following messages:

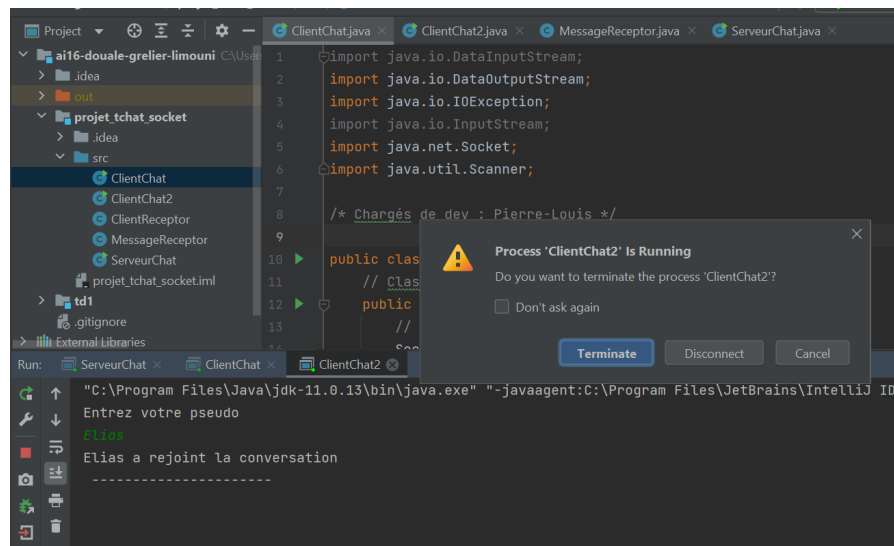
```
Elias: Elias
Elias a quitte la conversation.
Elias a quitte la conversation.
EXIT
Deconnexion du client

Process finished with exit code 0
```

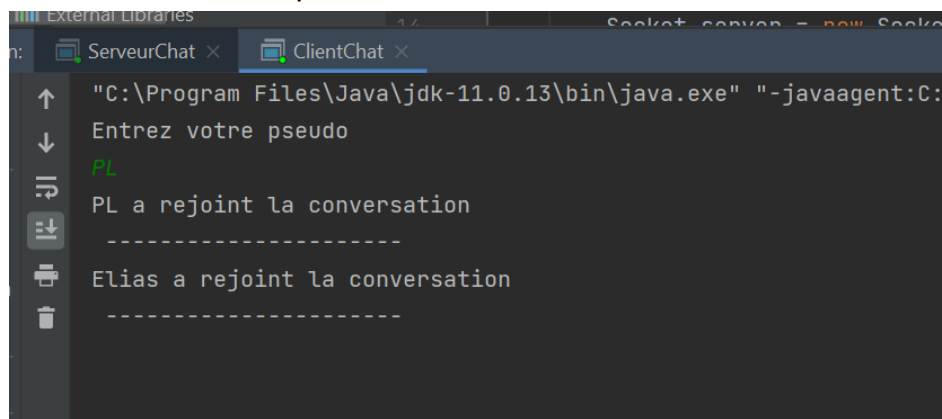
"Elias" ne reçoit bien évidemment pas la déconnexion de "PL" car il s'est déconnecté en premier.

Autres tests:

Cas où un utilisateur se déconnecte sans faire EXIT mais en fermant le terminal ("disconnect"):



Il n'y a rien sur le terminal du premier client:



Cas où ClientChat est lancé avant que le ServerChat ne soit lancé:

