TP RMI Polling

Dans cette partie du TP RMI, il fallait réaliser un chat en utilisant la méthode du Polling.

Donc ce projet java TPRMI_Polling est composé de 3 packages, le premier contient la classe client ainsi que les Thread PollThread et SendThread, le deuxième contient l'interface Chat et le dernier contient la classe Serveur.

Le Serveur implémente l'interface Chat. Il est composé de plusieurs méthodes et dont le rôle est de récupérer les messages envoyés de tous les clients et permet à ceux-ci de récupérer les messages leurs étant destinés.

Pour ce faire, il possède un main où j'instancie un Serveur et donne une adresse au serveur. La méthode envoieDeMessage permet de récupérer les messages des clients, et les ajoute à la liste des messages de tous les clients, et ajoute également le nom de l'utilisateur d'où proviennent les messages en parallèle de la liste des messages. C'est-à-dire que chaque message envoyé aura un nom d'utilisateur au même indice que lui dans la liste users. Cette distinction entre les messages permettra de ne pas afficher les messages à l'utilisateur qui les envoie. La méthode returnMessageList renvoie la liste des messages que le serveur possède, et la méthode returnUser permet de retourner les listes des utilisateurs, je veux parler de la liste « parallèle » à celle des messages.

La classe Client instancie un client et établit la connexion avec le serveur, elle possède également une liste des messages envoyés avec la méthodes returnMessageList qi permet de la retourner. Elle instancie les deux Thread, un pour envoyer les messages, l'autre pour afficher les messages reçu. De plus si la connexion avec le serveur est impossible, elle stop le client et l'éteint.

Le PollThread affiche les messages reçu. Pour se faire il récupère à chaque tour de boucle la liste des messages du client et celle du serveur, puis compare leur taille. Ainsi si la liste côté serveur est plus grande que celle du client, cela signifie que des messages ont été envoyés, et il les affiches tout en complétant sa liste des messages . De plus en récupérant la liste users « parallèle » de celle des messages du serveur, il n'affiche que les messages envoyés des autres clients. De plus si la connexion avec le serveur crash, elle stop le client et l'éteint.

Le SendThread envoie les messages du client au serveur qui les stocke. Pour ce faire il utilise la méthode envoieDeMessage du serveur. Quand le client entre « quit », il éteint le client.

TP RMI CallBacks

Dans cette partie du TP RMI, il fallait réaliser un chat en utilisant la méthode du CallBacks.

Donc ce projet java TPRMI_Callbacks est composé de trois packages, le premiers contient la classe Client, le deuxième, les interfaces Chat et Chat2 et le dernier la classe Serveur.

Cette fois ci le client ne va plus aller chercher les messages envoyés dans le Serveur, mais c'est celui-ci qui va les lui envoyer.

Pour ce faire la classe Serveur implémente une l'interface Chat. Elle possède plusieurs méthodes. La méthode envoieDeMessage qui envoie les messages accompagnés du pseudo du client qui en est l'auteur en faisant appel à la méthode returnMessage. Celle-ci fait appel à la méthode de la classe Client afficheMessage, qui affiche les messages reçu, pour chaque client présent dans la liste des clients, de plus un test permet de n'afficher ces messages qu'aux clients qui n'en sont pas les auteurs. Cette classe instancie un Serveur et donne une adresse au serveur (dans le main).

La classe Client instancie cette fois une nouvelle interface, l'interface Chat2. Elle instancie un client, l'ajoute à la liste des clients , se connecte au serveur et demande un pseudo à l'utilisateur, si la connexion est impossible, elle éteint le client (après envoie un message pour prévenir) et l'enlève de la liste des clients. De plus quand l'utilisateur entre « quit » après envoie d'un message pour prévenir elle éteint le client et l'enlève également de la liste des clients. Cette liste est présente dans le serveur.

Le serveur ne stocke pas les messages dans cette version.