Rapport de Travaux Pratique sur RMI

Dans ce rapport, je vais expliquer le fonctionnement de mon « chat » en RMI. Ma version la plus avancée est la version avec les « Callback ». La version « Polling » fonctionne mais avec moins de fonctionnalités.

Le projet Callback est contenu dans le dossier « RMICallback ».

Pour commencer, il faut lancer le serveur en premier puis les clients. Il y a aucune restriction sur le moment où se connecte un client. Les clients, après avoir rentré un pseudo, peuvent envoyer des messages ou quitter en écrivant « /quit » dans la console.

Ma version du chat en RMI contient plusieurs fonctionnalités :

- Afficher l'historique des messages lors de la connexion d'un client.
- Lors de l'envoie d'un message tous les clients reçoivent le message sauf l'émetteur.
- Un message s'affiche lors de la connexion ou la déconnexion d'un client.
- Lors d'une erreur de communication entre le client et le serveur, le serveur supprime le client en question de sa liste des clients afin de relever le problème qu'une seule fois sans altérer la communication entre les autres clients.

Pour la partie affichage de l'historique, cela se passe dans la méthode connexion. Cela envoie au client qui vient de se connecter les messages précédents dans l'ordre. En plus, elle permet de diffuser le fait qu'un nouveau client vient de se connecter.

Pour l'envoie d'un message, il a fallu que je rajoute le client émetteur du message en argument de ma méthode d'envoie afin de comparer cela aux autres clients de la liste sur le serveur. Cela permet de ne pas envoyer le message à l'émetteur.

La gestion d'une erreur lors d'un envoi ou d'une réception de message à un client se fait en supprimant le client source de l'erreur de la liste des clients dans le serveur. J'ai remarqué que si on laissait l'erreur, le serveur tournait beaucoup plus lentement mais en conservant ses fonctionnalités. D'où mon choix de tous de suite déconnecter le client dysfonctionnant.

Lors d'une déconnexion normale (« /quit »), le client demande une déconnexion au serveur afin de ne plus recevoir de message et dans un deuxième temps, la boucle « while » permettant l'envoi se termine du côté du client afin d'arrêter la possibilité d'envoi de message.