AUDIFFRET Sam

TP2 : RMI

RMIChatV2 :

Chat RMI en utilisant des callbacks.

Un client s'initialise en se donnant un nom, se connecte au serveur puis au chat, il peut ensuite envoyer des messages. Lorsque le client écrit « quit », il se déconnecte du chat et se ferme.

Quand un client se connecte au chat, il appel la méthode « arrive » de l'objet serveur qui l'ajoute à la liste des clients connecté et ajoute son nom à la liste des noms des clients connectés puis appel la méthode « annonce » du serveur pour annoncer aux clients connectés la nouvelle connexion.

La méthode « annonce » du serveur parcourt la liste des clients connectés et appel la méthode « message » de chaque client pour afficher le message d'annonce dans la console du client. Si le serveur ne parvient pas à envoyer un message à un client, il appel la méthode « crash » du serveur.

La méthode « message » du client affiche un message donné dans la console du client.

La méthode « crash » du serveur stock le nom du client donné dans une variable « nom » en le récupérant depuis la liste des noms stocké sur le serveur, retire le client de la liste des clients et retire le nom du client de la liste des noms, puis appel la méthode « annonce » du serveur pour annoncer aux clients qu'un client n'est plus joignable en précisant son nom que l'on a stocké dans la variable « nom ».

Quand un client envoi un message, il appel la méthode « ajoutMessage » de l'objet serveur pour le diffuser aux autres clients connectés. La méthode « ajoutMessage » parcourt la liste des clients connectés et compare chaque client au client qui a envoyé un message, lorsque ces deux derniers ne sont pas égaux, il appel la méthode « message » du client pour afficher un message contenant le message envoyé ainsi que le nom du client qui l'a envoyé en appelant la méthode « getNom » du client ayant envoyé le message. Si le serveur ne parvient pas à appeler la méthode « message » d'un client, il appel la méthode « crash » du serveur.

Quand un client se déconnecte, il appel la méthode « depart » de l'objet serveur qui le retire du chat. La méthode « depart » retire le client de la liste des clients connecté puis appel la méthode d'annonce du serveur pour annoncer aux clients connectés la déconnexion.

Le serveur ne conserve pas les messages, cela évite d'éventuel problèmes de mémoire s'il arrive un moment où il y a eu énormément de messages envoyés. Si l'on veut pouvoir superviser le chat, le code est facilement modifiable pour conserver les messages : il suffit de créer une ArrayList et d'y ajouter chaque message envoyé dans la méthode « AjoutMessage ».

Le serveur conserve les noms des clients connectés pour pouvoir savoir quel client n'est plus accessible en cas de déconnection inattendue (dans quel cas le client ne serait plus accessible, on ne pourrait donc plus appeler sa méthode « getNom »).

Lorsqu'un client se déconnecte de façon inattendue, le serveur s'en rendra compte si un client envoie un message ou si un nouveau client se connecte mais cela n'est pas immédiat : si un autre message est envoyé ou si un autre client se connecte avant que le serveur s'en rende compte, une erreur apparaîtra car il essaiera de retirer le client qui a crashé de la liste des clients plusieurs fois. Le chat restera cependant fonctionnel.