TP ARAR - Communication Serveur-Client en Java

VILLERMET Quentin 11507338 SUBLET Gary 11506450

Fonctionnalités Développées

- Le *Serveur* se lance et, grâce à un *DatagramSocket*, écoute sur un port et une adresse IP par défaut (modifiables directement dans le code source).
- Le *Client* se lance, crée lui aussi un *DatagramSocket*, et envoie un message de début de communication au Serveur.
- Lors de son lancement, le *Client* ouvre aussi une nouvelle *CommunicationThread* (en mode "client") qui sera chargée d'écouter les réponses du *Serveur* et de les afficher dans la console du *Client*. Cela permet de créer une "chatroom", puisque tous les messages envoyés par les clients seront broadcastés à l'ensemble des clients connectés par le biais de cette thread d'écoute.
- Lorsque le *Serveur* reçoit un message, il vérifie son contenu. Si le message est un message de début de connection (i.e. de type "hello rx302"), celui-ci détecte alors une nouvelle connection, et ouvre en conséquence une nouvelle *CommunicationThread* (cette foic-ci en mode "serveur"), qui aura pour rôle de recevoir les messages du *Client* avec laquelle elle est alors associée, et uniquement ceux de ce *Client*. Il est à noter que cette même *CommunicationThread* a aussi pour rôle d'ajouter son *Client* à la liste des clients connectés. Cette liste sera utilisée ensuite lors des broadcats de messages aux clients. La *CommunicationThread* "serveur" réceptionne les messages venant de son *Client*, et les broadcaste dans la "chatroom". Elle fonctionne comme un mini-serveur indépendant.
- Ces 3 classes héritent toutes de *Com*, qui est notre classe abstraite maîtresse. On y trouve certaines variables statiques comme l'IP et le port par défaut, mais aussi des méthodes cruciales et généralistes comme *send()* pour envoyer des messages, et *scan()* qui permet de trouver un port libre sur lequel créer un *DatagramSocket*.
- En pratique, nous avons donc un serveur de type chatroom concurrent UDP, permettant à plusieurs utilisateurs de se connecter et de communiquer ensemble, avec des traitements côté serveur effectués en parallèle.

Capture d'écran - Mise en situation

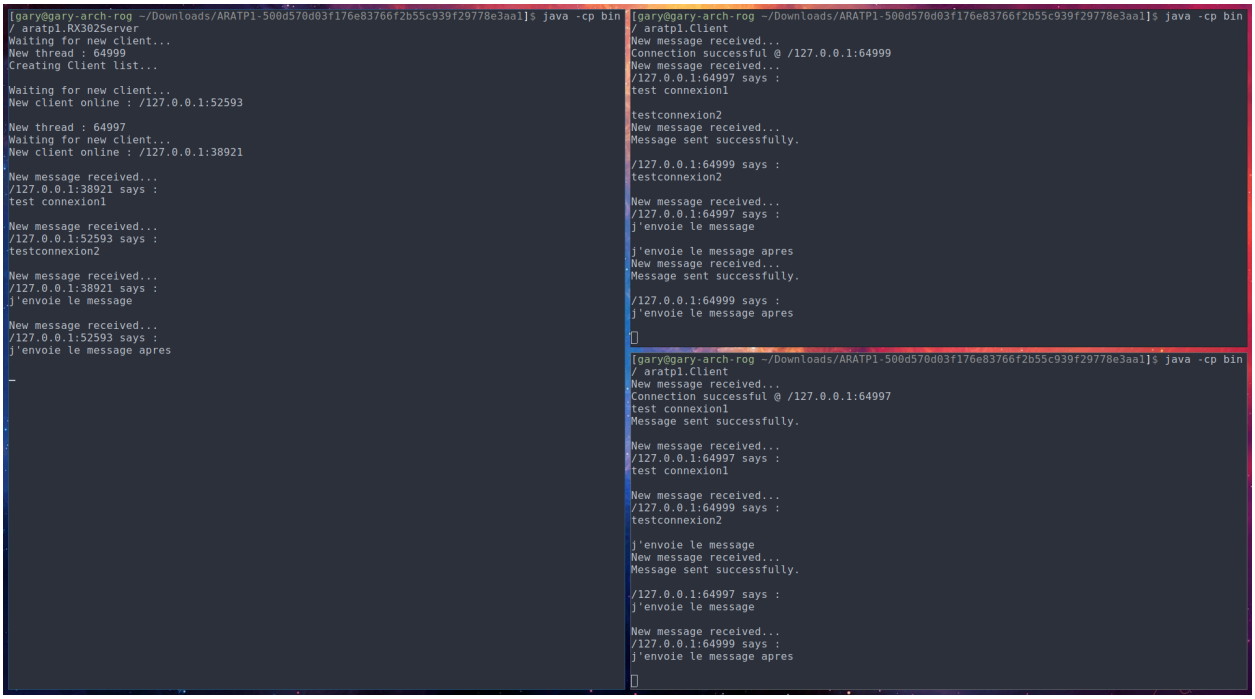

Diagramme UML