Programmation événementielle
TP 3 – Les sockets (objets)

Objectif

L'objectif de ce TP est de mettre en place une application client-serveur échangeant des objets (car dans le précédent TP, vos applications ont seulement échangé des chaines de caractères). Plus précisément, chaque objet transmis du client au serveur contiendra une phrase et un compteur incrémenté à chaque envoi.

Création de la classe Message

1. Préparez un environnement de travail avec un package tp7 qui contiendra l'ensemble du code source de ce TP.
2. Créez une classe Message qui sera utilisée pour les messages envoyés par les clients. La classe Message devra contenir un attribut « text » contenant le texte à transmettre et un attribut « number » correspondant au numéro du message.
3. Implémentez la méthode toString() de la classe Message.
4. Implémentez l'interface Serializable pour la classe Message. Cette interface permet d'envoyer un objet sous la forme d'un flux sérialisé. Cela est utile pour sauvegarder un objet sur un disque ou… pour transmettre un objet entre un client et un serveur.
5. Ajoutez une méthode read() qui lit sur l'entrée standard une chaine de caractères et l'affecte à l'attribut text.
6. Testez votre classe Message en créant un objet, en appelant la méthode read() et en affichant l'objet.

Création du serveur et du client basiques

7. Créez une classe MessageServer qui recevra les connexion TCP/IP sur le port 10000.
8. Implémentez la réception d'objets de la classe Message. A chaque réception, le serveur affichera le texte transmis et le numéro du message. Pour cela, vous pouvez vous appuyer sur :
   - La classe ObjectInputStream qui sert à transformer un flux d'entrée InputStream en un flux d'objets. Pour bénéficier de cette classe, vous pouvez vous inspirer des précédents TP en l'utilisant à la place de la classe InputStreamReader qui est dédiée uniquement aux chaines de caractères.
   - La méthode readObject() de la classe ObjectInputStream pour lire l'objet. Comme l'objet lu est de la classe Object, il faut forcer le typage Message avec un cast.
9. Créez une classe MessageClient qui se connectera au serveur sur le port 10000.
10. Envoyez des messages au serveur écrits par l'utilisateur sur l'entrée standard. A chaque nouvel envoi, le compteur sera incrémenté. De manière symétrique par rapport au serveur, il faut exploiter la clase ObjectOutputStream et sa méthode writeObject(Object object).
11. Testez vos classes en démarrant d'une part un processus serveur avec un objet de la classe MessageServer et d'autre part, un processus client avec un objet de la classe MessageClient.

Création du serveur et du client avancés

12. Modifiez votre serveur pour qu'il puisse accepter plusieurs clients.
13. Faites en sorte que la saisie du texte « exit » arrête le client concerné et que la saisie du texte « kill » arrête le client et le serveur.