PR6 – Programmation réseaux
TP n° 5 : Producteurs-Consommateurs en réseaux

Exercice 1 : un producteur et un consommateur

Le but de cet exercice est de programmer une application réseau avec un client producteur et un client consommateur. Pour ce faire, le serveur écoute sur deux ports. Sur un port, le serveur attend les connexions d'un client producteur et sur l'autre port les connexions d'un client consommateur. L'idée est la suivante : un client producteur envoie au serveur une chaîne de caractères qu'il aura reçu par l'entrée standard (le clavier), le serveur stocke alors cette chaîne de caractères dans un tampon (buffer) `buf`. Quant au client consommateur il attend que le serveur lui envoie des chaînes de caractères qui ont été placés dans le tampon. Il faudra donc programmer deux types de client `ClientCons` et `ClientProd` et un serveur `Serv`.
Afin que le serveur attende des connexions sur deux ports différents, vous pouvez programmer un serveur multi-threadé. Les threads du serveur devront alors partager le tampon de chaînes de caractères `buf`. Pour éviter les accès concurrents à `buf`, il suffit de créer `buf` comme un objet de type `CopyOnWriteArrayList<String>` du package `java.util.concurrent`. Un objet de type `CopyOnWriteArrayList<String>` est équivalent à une `ArrayList` mais avec gestion des accès concurrents.
Pour tester ensuite votre programme, vous pouvez réaliser les étapes suivantes :

1. lancer le serveur,
2. puis lancer le client consommateur et sauvegardez la sortie dans un fichier :
   `java ClientCons > testClient.txt`
3. lancer le client producteur en lui faisant envoyer une centaine de messages :
   `(for i in {1..100}; do echo "bla bla"; done; echo .) | java ClientProd`
4. vérifiez que le fichier `testClient.txt` contient l'intégralité des messages.

Exercice 2 : plusieurs producteurs et plusieurs consommateurs

Modifier le programme de l'exercice précédent afin de gérer plusieurs producteurs et plusieurs consommateurs. Il faut réfléchir à comment faire en multi-threadé et à quel moment créer les différents threads et quels sont les objets partagés par ces threads.