PR6 – Programmation réseaux
TP n° 6 : Clients et Serveurs TCP en Java

N'oubliez pas qu'en Java, vous **devez** rattraper les exceptions. Ici, il s'agira exclusivement d'exceptions du type `IOException`.
Remarque : Dans le document le signe ␣ représentera un simple caractère d'espacement (ASCII 32). De plus les messages circulant sont indiqués entre guillemets, **les guillemets ne faisant pas partie du message**.

Exercice 1 : Client `daytime` en Java

Écrire un programme qui se connecte au service `daytime` sur `lampe` et affiche la date et l'heure renvoyées par le service et se déconnecte. On fera une version où l'on spécifiera directement l'adresse IP de `lampe` et une autre où on laissera Java la résoudre à partir de `lampe`.

Exercice 2 : Serveur `echo` en Java

Écrire un serveur TCP implémentant le service `echo` (pour tout client se connectant, le serveur répète au client tout ce que celui-ci lui envoie). Le serveur devra pouvoir traiter un nombre quelconque de clients *à la suite*. Tester le serveur en utilisant `nc` comme client.

Exercice 3 : Jeu en Java

Dans cet exercice, on implémente un serveur en Java qui exécute le jeu déjà vu plusieurs fois que l'on rappelle néanmoins ci-après. Lorsqu'un client se connecte, le serveur choisit au hasard un nombre `n` entre 0 et 65535. Le client peut alors faire jusqu'à 20 tentatives pour deviner `n` en envoyant des requêtes de la forme `"k\n"` au serveur, où `k` est un entier compris entre 0 et 65535 représenté comme chaîne de caractères. À chaque requête, le serveur répond :

- `"PLUS␣r\n"` si `k` inférieur à `n`, où `r` est le nombre de tentatives restantes ;
- `"MOINS␣r\n"` si `k` est supérieur à `n`, où `r` est le nombre de tentatives restantes ;
- `"GAGNE\n"` si `k` est égal à `n`. Le serveur clôt alors la communication ;
- `"PERDU\n"` si `k` est différent de `n` et qu'il ne reste plus de tentative au client. Le serveur clôt alors la communication.

1. Implémenter un serveur qui exécute le protocole ci-dessus.

2. Tester le serveur en utilisant `nc` et le client que vous avez écrit en C.

3. Écrire un client qui exécute automatiquement une partie avec le serveur et le tester avec le serveur écrit en C et celui en Java.

4. Écrire un programme en Java qui lance 100 clients en parallèle avec des threads. L'exécuter avec le serveur en C pour l'exercice 1 du TP4. Est-ce que les différents clients sont bien traités en parallèle ?

5. Adapter le code du serveur en Java afin qu'il puisse traiter les clients en parallèle comme dans la question 1 du TP 4.

6. Écrire un autre serveur en Java qui permet des parties à deux joueurs suivant le protocole de la question 2 de l'exercice 1 du TP4. Permettre l'exécution de parties en parallèle. Adapter le code de la question 4 afin de pouvoir tester ce serveur.