TD5 Java

Les entrées-sorties

Exercice 1

Lire ce fichier **texte.txt** en utilisant la classe **java.io.FileReader**.

Exercice 2

Lire ce fichier **texte.txt** en utilisant la classe **java.io.BufferedReader**.

Exercice 3

Comparez le temps d'exécution de la lecture du fichier **texte.txt** en utilisant d'abord la librairie **java.io**, puis la librairie **java.nio**.
Vous pouvez utiliser la fonction `System.currentTimeMillis()` pour calculer le temps d'exécution.

Exercice 4

Créer une classe Client et une classe Serveur avec permettant de faire une connexion sur le port 2000.

Exercice 5

Modifiez les classes Client et Serveur afin d'échanger un message entre le client et le serveur.

- Côté serveur :
  - Le serveur est en attente d'un flux :
    `out = PrintWriter(socket.getOutputStream);`
  - Dès que la connexion est établie, le serveur prépare l'envoi d'un message aux clients :
    `out.println("Message");`
  - Le serveur envoie le message : `out.flush();`
- Côté client :
  - Le client se met en attente de reception du message :
    `BufferedReader(new InputStreamReader(socket.getInputStream());`
  - Le client lis le message et l'affiche dans la console : `in.readLine();`

Exercice 6

Écrire un programme Java qui génère 10 processus. Chaque processus devra patienter aléatoirement entre 0 et 10 secondes, puis afficher un message indiquant que le processus a bien été exécuté.

Exercice 7

Modifier notre serveur afin que chaque requête d'un client soit implémentée dans un processus.
Pour ceci, nous implémenterons une classe `AcceptClient` dont le constructeur prend en paramètres une Socket.
Le serveur restera en activité en permanence grâce à un `while(true)` et devra retourner le nombre de cleints qui se sont connectés.

Exercice 8

En s'inspirant des classes Client et Serveur précédentes, faire un service de messagerie.
La classe `AcceptClient` permet de gérer la connexion avec un nouvel utilisateur.
La classe `ReceiveMessage` permet au serveur de récupérer les messages des clients et de les afficher dans la console.
La classe `BroadcastMessage` permet au serveur de redistribuer à l'ensemble des clients le message reçu.

Bonus : faire un système de nom d'utilisateur permettant de savoir qui a envoyé le message.