

Examen de Langage Java

Durée : 1h

Aucun document autorisé

Mobiles interdits Note : la qualité des commentaires, avec notamment la présence d'affirmations significatives, ainsi que les noms donnés aux variables, et la bonne indentation rentrent pour une part importante dans l'appréciation du travail.

Vous choisirez de répondre à UNE question parmi les 2 suivantes :

Question 1

- 1. On désire calculer la somme de tous les éléments d'une matrice de réels double. Pour cela, on calculera la somme des éléments de chacune des lignes de la matrice, puis sommerá les sommes par ligne pour obtenir la somme totale.
Le calcul de la somme des éléments d'une ligne sera effectué à l'aide d'un thread. Il y aura donc autant de threads que de lignes que comporte la matrice.
- 2. Écrivez la classe `SommeLigne` qui définit un thread calculant la somme des réels d'une ligne d'une matrice.
- 3. Écrivez la classe `Matrice`, qui comporte la méthode `main` qui calcule la somme des éléments d'une matrice de réels (que vous déclarerez) en utilisant des threads du type `SommeLigne` que vous avez précédemment défini.

Question 2

- 4. On souhaite écrire un serveur TCP pour faire des calculs sur des entiers avec les quatre opérations $+$, $-$, $\times$ et $/$. Le client envoie au serveur des commandes sur une ligne de la forme « opération entier » (e.g. $+ 6$ ou $/ -3$). Le serveur envoie en retour le résultat de l'opération. Les opérations sont réalisées séquentiellement avec une initialisation à 0 ; tous les opérateurs ont la même priorité et sont associatifs à gauche. La commande `fin` termine les calculs. Voici un exemple d'échanges entre un client et le serveur qui correspond à l'opération $((0+20)-5)/3$:

```
Client> + 20
Serveur> 20
Client> - 5
Serveur> 15
Client> / 3
Serveur> 5
Client> fin
Serveur> Fin connexion
```

- 5. Écrivez la méthode `main` de la classe `Serveur` qui accepte les connexions de clients qui lui présentent des opérateurs arithmétiques à réaliser.
- 6. Écrivez la classe `GestionDuClient` (un thread) qui réalise sur le serveur les opérations de calcul communiquées par un client et qui lui renvoie les résultats. L'objet `s` représente une socket de communication TCP déjà ouverte vers un client.

Note : vous pourrez utiliser la méthode `substring(int i, int j)` de `String` qui renvoie une sous-chaine comprise entre les indices `i` (inclus) et `j` (exclus). D'autre part, un client pourra être représenté par la classe suivante :

```
/*
 * Usage : java Client serveur port
 */
import java.net.*;
import java.io.*;

public class Client {
    public static void main (String [] args) {
        int portDest = Integer.parseInt(args[1]);
        String serveur = args[0];
        //
        try (
            Socket socket = new Socket(serveur, portDest);
            PrintStream out = new PrintStream(socket.getOutputStream());
            BufferedReader in = new BufferedReader(
                new InputStreamReader(socket.getInputStream()))
        {
            Console console;
            if ((console = System.console()) == null) {
                System.err.println("Erreur : pas de console disponible");
                System.exit(1);
            }
            boolean fin=false;
            do {
                String cmd = console.readLine("client>");
                out.println(cmd);
                String réponse = in.readLine();
                if (réponse.equals("fin")) fin=true;
                else
                    System.out.println(réponse);
            } while (!fin);
        }
        catch (IOException e) {
            System.err.println(": "+ e);
        }
    }
}
```