Réseaux avancés
M1 GL 2023/2024 - TP N° 4 (Sockets TCP avec IHM)

Sommaire

Exercice 1 : Communication JFrame/Serveur avec TCP 1
Exercice 2 : Opérations mathématiques avec IHM 3
Exercice 3 : Calculatrice client/serveur avec TCP 5

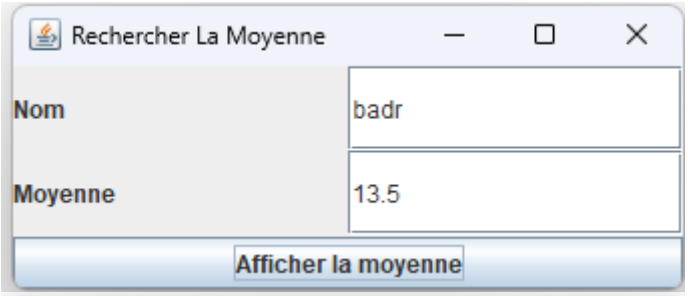
Exercice 1 : Communication JFrame/Serveur avec TCP

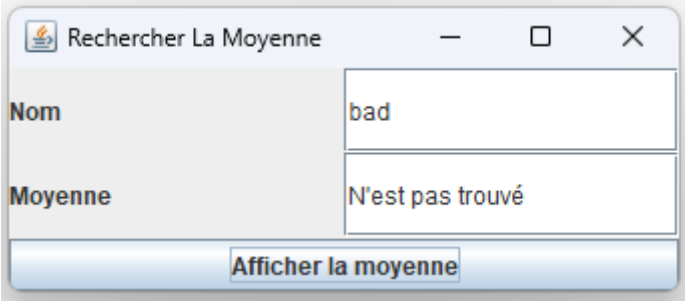
Soit les deux classes Java suivante :

```
import java.net.*;
import java.io.*;
import java.util.HashMap;

class Serveur {
private HashMap<String, Double> table;

public Serveur() {
table =new HashMap<>();
table.put("badr", 13.5);
table.put("saleh", 15.5);
table.put("youcef", 9.5);
table.put("anes", 8.5);
 }

  public double TrouverMoy (String name) {
  if (table.containsKey(name))   return (table.get(name));
  else return -1;
 }

   public static void main(String args[]) {
   Serveur b = new Serveur();
   System.out.println("Serveur en attente de connexion ...");
   ServerSocket server = null;
   try {
     server = new ServerSocket(7777);
     while (true) {
       Socket sock = server.accept();
       System.out.println("connecte");
       DataOutputStream sockOut = new DataOutputStream(sock.getOutputStream());
       DataInputStream sockIn = new DataInputStream(sock.getInputStream());
       String name;

       while ((name = sockIn.readUTF()) != null)  {
          sockOut.writeDouble(b.TrouverMoy (name));
          sockOut.flush();
       }

       sockOut.close();
       sock.close();
     }
 } catch (IOException e) {try {server.close();} catch (IOException e2) {}}
}
 }
```

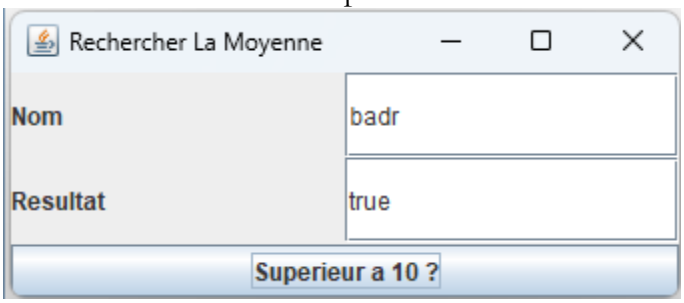
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.net.*;

public class Client extends JFrame implements ActionListener {
Socket sock = null;
DataOutputStream sockOut = null;
DataInputStream sockIn = null;
private JButton jbtGetMoy = new JButton("Afficher la moyenne");
private JTextField jtfNom = new JTextField();
private JTextField jtfMoy = new JTextField();

public Client(){
JPanel panneau = new JPanel();
panneau.setLayout(new GridLayout(2, 2));
panneau.add(new JLabel("Nom"));
panneau.add(jtfNom);
panneau.add(new JLabel("Moyenne"));
panneau.add(jtfMoy);
add(panneau, BorderLayout.CENTER);
add(jbtGetMoy, BorderLayout.SOUTH);
jbtGetMoy.addActionListener (this);
}

public void init() {
   try {sock = new Socket("localhost", 7777);
     sockOut = new DataOutputStream(sock.getOutputStream());
     sockIn = new DataInputStream(sock.getInputStream());
   } catch (UnknownHostException e) {
     System.err.println("host non atteignable : localhost");
     System.exit(1);
   } catch (IOException e) {
     System.err.println("connection impossible avec : localhost");
     System.exit(1);
   }
 }


 @Override
 public void actionPerformed (ActionEvent e) {
 double score=0;
 try {
 sockOut.writeUTF(jtfNom.getText().trim());
 sockOut.flush();
} catch (IOException ex) {}
 try {score = sockIn.readDouble();} catch (IOException ex) {}
if (score < 0) jtfMoy.setText("N'est pas trouvé");
 else jtfMoy.setText(Double.toString(score));
 }

public static void main(String[] args) {
Client a = new Client();
a.setTitle("Rechercher La Moyenne");
a.setSize(350, 150);
a.init();
a.setLocationRelativeTo(null);
a.setVisible(true);
 }
}
```

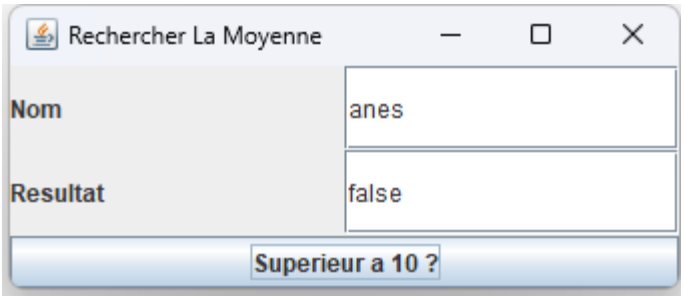
Une éventuelle exécution est comme suit :





Modifier les deux classes précédentes afin d'avoir une éventuelle exécution comme suit :



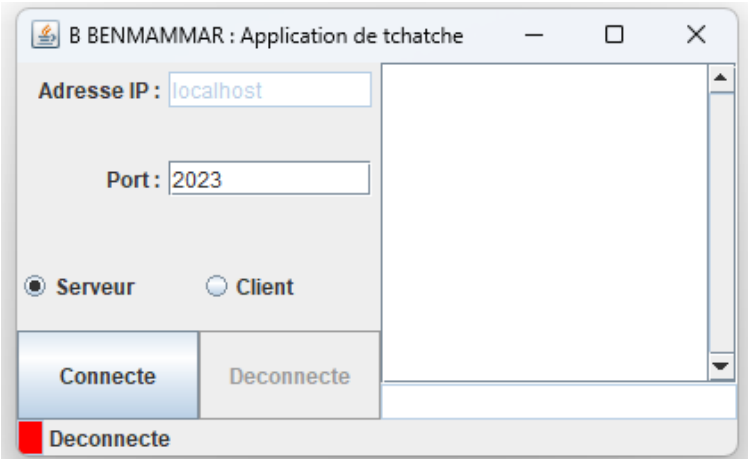


En passant le nom d'un étudiant à partir du JFrame, le serveur répond par un booléen.

- true : si a moyenne est supérieure ou égale a 10.
- false : si sa moyenne est inférieure à 10 ou si son nom n'existe pas dans le serveur.

Exercice 2 : Opérations mathématiques avec IHM

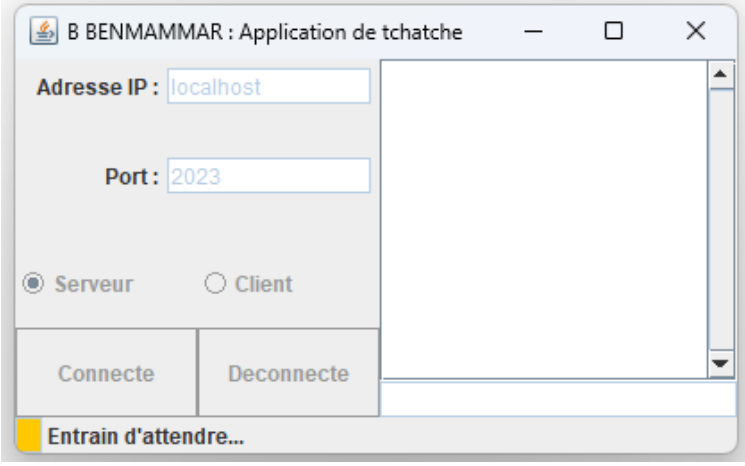
Lancer l'exécution de **TCPChat.java**. Il s'agit d'une application de tchatche qui est réalisée avec la bibliothèque swing et les sockets TCP. C'est une seule classe qui représente à la fois le client est le serveur. Au lancement de l'application la fenêtre **JFrame** suivante est apparue (combinaison de **JLabel, JTextField, JPanel, JTextArea, JRadioButton, ...)** :



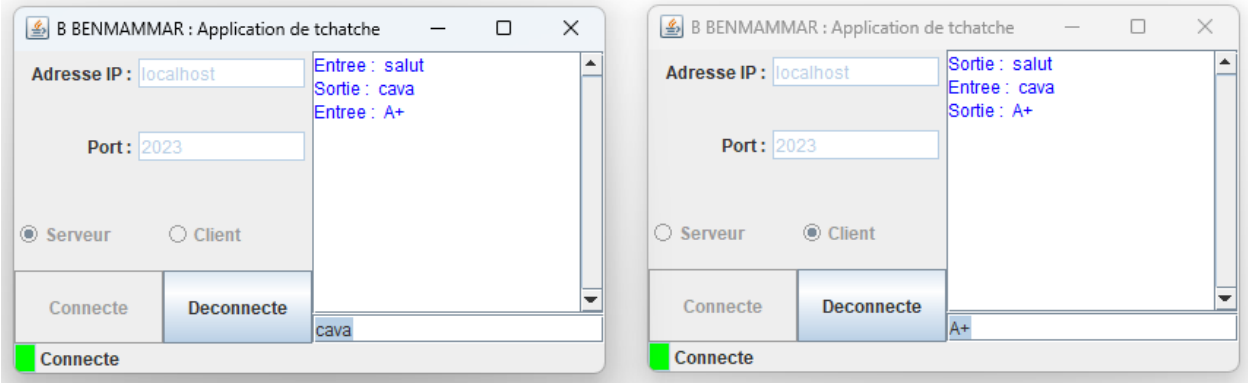
Pour tchatcher il faut lancer deux instances de cette application, une instance pour la première personne qui doit choisir le ***bouton Serveur*** et une instance pour la deuxième personne qui doit choisir le ***bouton Client***. A l'intérieur de la classe **TCPChat**, on peut trouver un code qui ressemble au suivant :

```
if (Serveur) {
        hostServer = new ServerSocket (port);
        socket = hostServer.accept(); // attente client
      }
      else { // si Client
        socket = new Socket ("localhost", port);
      }
```

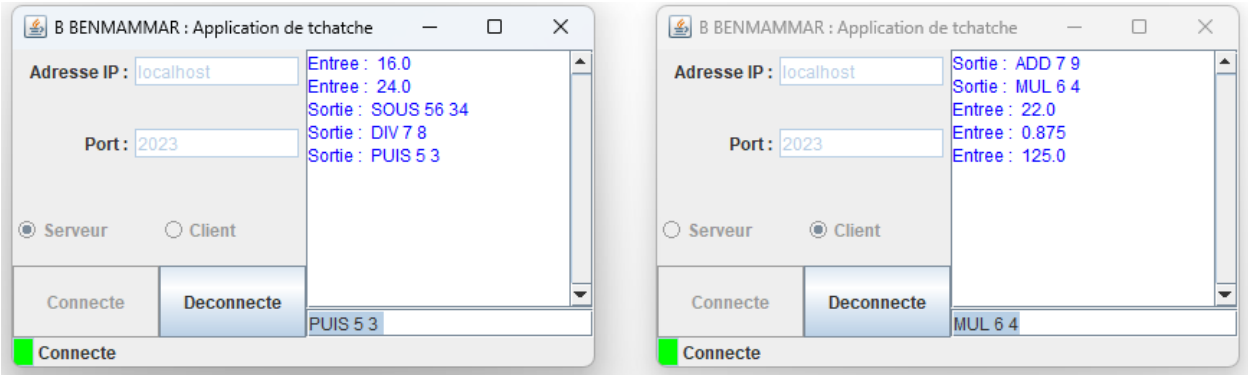
Avant la connexion du client (2ème personne, on a la vue suivante coté serveur) :



Après la connexion du client, les deux personnes peuvent échanger leurs messages :



Votre travail consiste à faire quelques modifications sur le code précédent afin de réaliser les opérations arithmétiques ADD, MUL, SOUS, DIV et PUIS.
Les IHMs attendues sont comme suit :



Par exemple, le serveur envoie dans son IHM (PUIS 5 3), le client verra dans le sien (Entree : 125.0).
Le client envoie dans son IHM (MUL 6 4), le serveur verra dans le sien (Entree : 24.0).

Dans la classe **TCPChat.java,** il faut se focaliser sur la partie suivante du code :

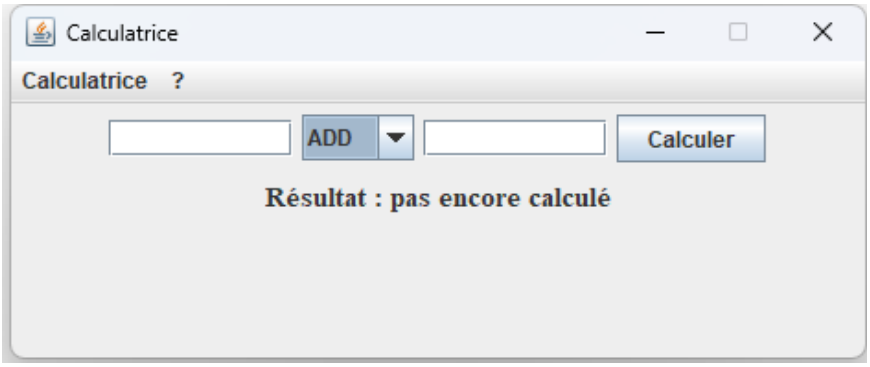
```
if (in.ready()) {
            s = in.readLine();
            if ((s != null) &&  (s.length() != 0)) {
              if (s.equals(END_CHAT_SESSION)) {
                changeStatusTS(DISCONNECTING, true);
              }
                else {
                 appendToChatBox("Entree :  " + s + "\n");
                 changeStatusTS(NULL, true);
              }
            }
          }
```

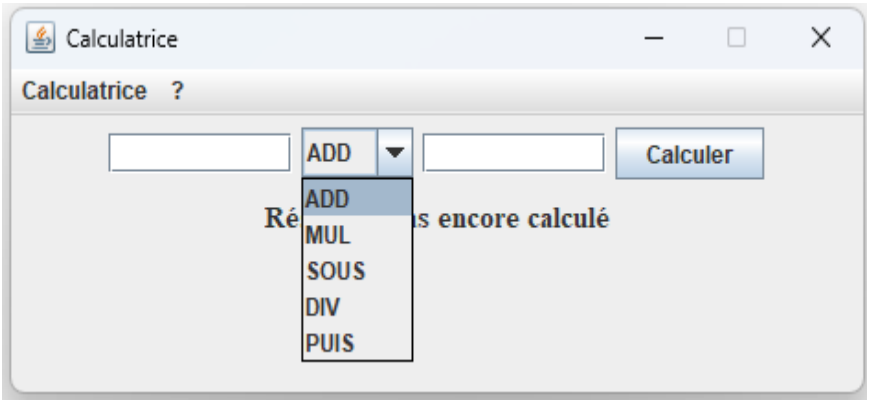
Exercice 3 : Calculatrice client/serveur avec TCP

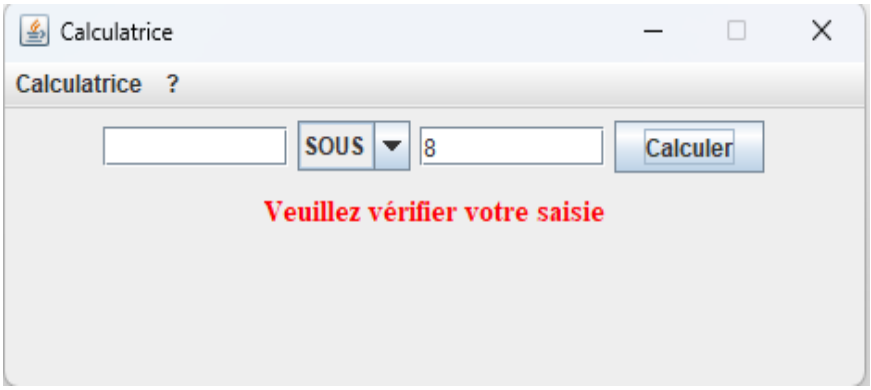
L'objectif de cet exercice est de développer une calculatrice client/serveur en se basant sur les sockets TCP. Dans le client sous forme de **JFrame,** l'utilisateur saisira les deux opérandes et choisira une opération (**ADD, MUL, DIV, SOUS ou PUIS**). Le calcul se fera par la suite coté serveur et l'affichage se fera coté client (dans un **JLabel** par exemple).
Lancer la classe Java suivante **(Server.java).** Elle permet d'établir la connexion avec la calculatrice mais les opérations ne sont pas encore implémentées pour le moment.

```
import java.io.*;
import java.net.*;
public class Server {
   public static void main(String[] args) throws IOException {
      ServerSocket server = new ServerSocket(6666);
      while (true) {
         Socket socket = server.accept();
         DataOutputStream dos = new DataOutputStream(socket.getOutputStream());
         DataInputStream dis = new DataInputStream(socket.getInputStream());
         String requete = "";
         while ((requete = dis.readUTF()) != null) {
            // code a ajouter
         }
      }
   }
}
```

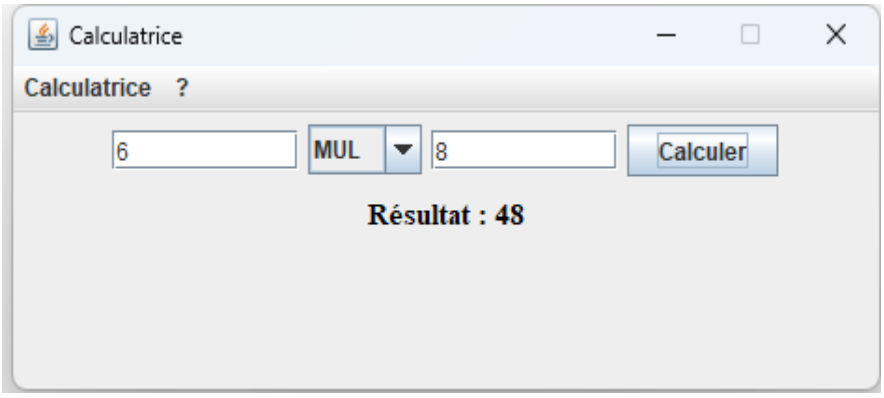
Lancer l'exécution de **Calculatrice.java**. Les vues attendues sont comme suit :







Votre objectif est de compléter le code du serveur afin d'implémenter toutes les opérations.
Une éventuelle vue sera comme suit :



Indication : utiliser la méthode **String.valueOf** afin de convertir le résultat de l'opération coté serveur en **String** pour l'envoyer au client avec la méthode **writeUTF** de **DataOutputStream.**