
Réseaux Avancés

M1 GL 2023/2024 - TP N° 5 (Sockets TCP)

Exercice 1 : Communication JFrame/Serveur avec TCP	1
Exercice 2 : Transfert d'images avec TCP	2
Exercice 3 : Transfert de données avec TCP	4

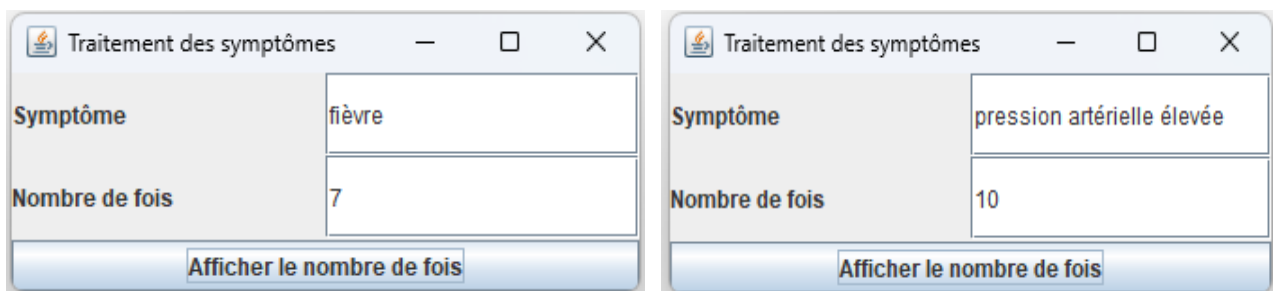
Exercice 1 : Communication JFrame/Serveur avec TCP

Avant de commencer, examiner le fichier : **symptomes.txt**. Il s'agit d'un contenu représentant 100 symptômes d'un certain nombre de patients. Un patient peut avoir un ou plusieurs symptômes.

Soit la classe java suivante :

```
import java.io.*;
import java.util.*;
public class A {
    public static void main(String args[]) throws IOException {
        HashMap<String, Integer> map = new HashMap<>();
        try (Scanner a = new Scanner(new File("./src/symptomes.txt"))) {
            while (a.hasNextLine()) ajouter(map, a.nextLine());
        }
    }
    static void ajouter(Map<String, Integer> map, String symptome) {
        map.compute(symptome, (key, value) -> (value == null) ? 1 : value + 1);
    }
}
```

1. Que fait la classe **A** ?
2. En s'appuyant sur l'exercice 1 du TP4, votre objectif est de réaliser l'IHM suivante coté client.



Pour le serveur, utiliser une structure de type **HashMap** (se baser sur la classe **A** précédente).

3. Dans ce qui suit, nous souhaiterons prendre en considération les noms des patients pour connaître leurs symptômes. Créer un autre fichier nommé **symptomesnom.txt** contenant ce qui suit :

Ali fièvre
Anes pupilles dilatées
Youcef bouche sèche
Ali inflammation
Ali tremblement
Anes douleur abdominale
Jaber pression artérielle élevée
Youcef nuque raide
Youcef toux

Votre objectif est de réaliser le traitement indiqué dans les IHMs suivantes :

The screenshot shows a window titled "Traitement des symptômes". It has two input fields: "Patient" with the value "Ali" and "Symptômes" with the value "fièvre inflammation tremblement". Below these fields is a button labeled "Afficher les symptômes".

The screenshot shows the same window "Traitement des symptômes". The "Patient" field now contains "Youcef" and the "Symptômes" field contains "bouche sèche nuque raide toux". The "Afficher les symptômes" button is still present at the bottom.

Exercice 2 : Transfert d'images avec TCP

1.1 Soit les deux classes java suivantes :

////////// ServerImage.java

```
import java.io.*;
import java.awt.image.BufferedImage;
import java.io.IOException;
import java.net.*;
import javax.imageio.ImageIO;
public class ServerImage{
    public static void main(String [] args) throws IOException{
        try{
            Socket server = new ServerSocket(6066).accept();
            BufferedImage
img=ImageIO.read(ImageIO.createImageInputStream(server.getInputStream()));
            ImageIO.write(img, "JPG", new File("./src/test.JPG"));
            System.out.println("Image reçue");
        } catch(Exception ex){}
    }
}
```

////////// ClientImage.java

```
import java.io.*;
import java.awt.image.BufferedImage;
import java.net.Socket;
import javax.imageio.ImageIO;
public class ClientImage{
    static BufferedImage bimg;
    public static void main(String [] args){
        try{
            Socket client = new Socket("localhost", 6066);
            bimg = ImageIO.read(new File("./src/univ.JPG"));
            ImageIO.write(bimg,"JPG",client.getOutputStream());
            client.close();
        }catch(IOException e){}
    }
}
```

- Avant de commencer, copier une image **"univ.JPG"** dans le répertoire contenant les sources **.java** (ex. une image liée à votre université). Exécuter les deux classes précédentes et examiner le résultat.
- Examiner la documentation de la classe **ImageIO** dans le lien suivant : <https://docs.oracle.com/javase/7/docs/api/javax/imageio/ImageIO.html>.
- Examiner le rôle des méthodes suivantes : **read** (**File** input), **write** (**RenderedImage** im, **String** formatName, **OutputStream** output), **createImageInputStream** (**Object** input), **read** (**InputStream** input) et **write** (**RenderedImage** im, **String** formatName, **File** output).

1.2 Soit les deux classes java suivantes :

////////// ClientCapt.java

```
import java.awt.*;
import java.awt.image.BufferedImage;
import java.net.Socket;
import javax.imageio.ImageIO;
public class ClientCapt{
    static BufferedImage image;
    public static void main(String [] args){
        try{
            Socket client = new Socket("localhost", 6000);
            Robot bot;
            bot = new Robot();
            image = bot.createScreenCapture(new Rectangle(0, 0, 500, 300));
            ImageIO.write(image,"JPG",client.getOutputStream());
            client.close();
        } catch(Exception e) {}
    }
}
```

////////// ServerCapt.java

```
import java.awt.HeadlessException;
import java.awt.image.BufferedImage;
import java.io.IOException;
import java.net.*;
import javax.imageio.ImageIO;
import javax.swing.*;
public class ServerCapt {
    public static void main(String [] args) throws Exception{
        ServerSocket serverSocket = new ServerSocket(6000);
        try{
            Socket server = serverSocket.accept();
            BufferedImage
img=ImageIO.read(ImageIO.createImageInputStream(server.getInputStream()));
            JFrame frame = new JFrame();
            frame.getContentPane().add(new JLabel(new ImageIcon(img)));
            frame.pack();
            frame.setVisible(true);
        }
        catch(IOException | HeadlessException e){}
    }
}
```

- Exécuter les deux classes précédentes. Que fait cette application ?
- Examiner le rôle des deux classes suivantes :
Robot (<https://docs.oracle.com/javase/7/docs/api/java/awt/Robot.html>) et
Rectangle (<https://docs.oracle.com/javase/7/docs/api/java/awt/Rectangle.html>).

Exercice 3 : Transfert de données avec TCP

- Que fait la méthode **mystere** de la classe Java suivante ?

```
import java.io.*;
public class ABC {
    public static void mystere (InputStream in, OutputStream out) throws IOException {
        byte buf[ ] = new byte[1024];
        int n;
        while((n=in.read(buf))!=-1) out.write(buf,0,n);
        in.close();
        out.close();
    }
}
```

- Soit les deux classes Java suivantes :

///////// Client.java

```
import java.io.*;
import java.net.*;
public class Client {
    public static void main(String []args) throws IOException {
        Socket sock = new Socket("localhost",9001);
        ABC.mystere(new FileInputStream("./src/test1.txt"),sock.getOutputStream());
        sock.close();
    }
}
```

/// Serveur.java

```
import java.io.*;
import java.net.*;
public class Serveur{
    public static void main(String []args) throws IOException {
        Socket sock = new ServerSocket(9001).accept();
        A.mystere(sock.getInputStream(),new FileOutputStream("./src/test2.txt"));
        sock.close();
    }
}
```

- Que fait le modèle client/serveur précédent ?
- Dans ce qui suit, nous souhaiterons donner au client la possibilité de choisir le type de la donnée qu'il souhaiterait envoyer au serveur.
- Examiner la classe java suivante :

```
import javax.imageio.ImageIO;
import java.awt.image.BufferedImage;
import java.io.*;
import java.net.Socket;
public class Client {
    public static void main(String[] args) throws IOException {
    Socket socket = new Socket("localhost", 9001);
    DataOutputStream out = new DataOutputStream(socket.getOutputStream());
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    System.out.println("Voulez-vous envoyer une image ou un fichier texte ? Tapez 'image' ou 'txt'");
    String type = br.readLine();
    out.writeUTF(type);
    if (type.equals("image")) {
        BufferedImage bimg = ImageIO.read(new File("./src/univ.jpg"));
        ImageIO.write(bimg, "JPG", socket.getOutputStream());
        System.out.println("Image envoyée");
    else if (type.equals("txt")) {
        ABC.mystere(new FileInputStream("./src/test1.txt"), socket.getOutputStream());
        System.out.println("Fichier texte envoyé");
    } out.close(); socket.close();
    }
}
```

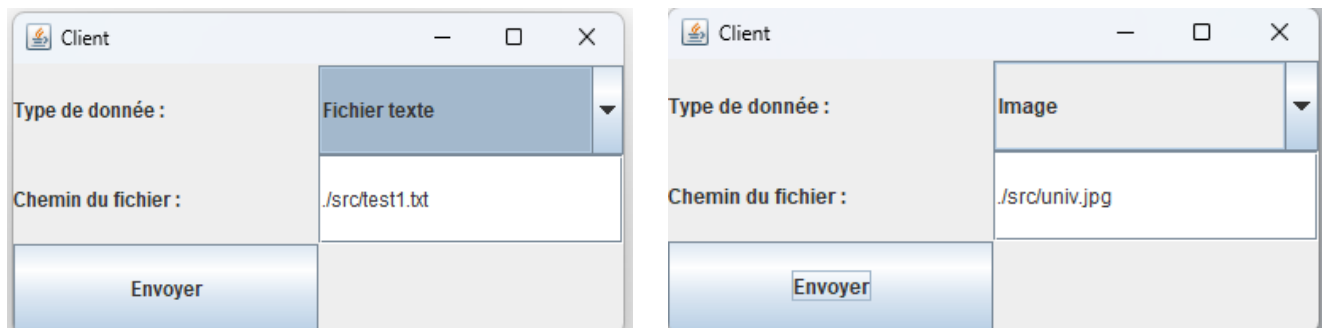
Compléter le code suivant du serveur :

```
import javax.imageio.ImageIO;
import java.awt.image.BufferedImage;
import java.io.*;
import java.net.*;

public class Serveur {
    public static void main(String[] args) throws IOException {
        ServerSocket serverSocket = new ServerSocket(9001);
        while (true) {
            Socket clientSocket = serverSocket.accept();
            .....
            .....
            if (type.equals("image")) {
                .....
                .....
                .....

            } else if (type.equals("txt")) {
                .....
                .....
            }
            clientSocket.close();
        }
    }
}
```

En gardant le code du serveur précédent, refaire le client sous forme d'IHM comme suit :



Améliorer votre application avec les fonctionnalités suivantes :

- La possibilité de connecter au même temps plusieurs clients au serveur en utilisant les threads.
- La possibilité d'authentifier les clients par le serveur avec « login/mot de passe ».
- La possibilité de récupérer une image ou un fichier à partir du serveur.