Tristan BRAU
Léo PHAV

Java pour les réseaux - TP 3

2.1 TCP Server

I watched this YouTube video to understand how to build a TCP client and server: *https://www.youtube.com/watch?v=gchR3DpY-8Q*. It helped me grasp the concepts of socket creation, stream handling, and message exchange over a reliable TCP connection.

```java
public class TCP_Server{
    Run | Debug
    public static void main(String[] args) throws IOException{
        Socket socket = null;
        InputStreamReader inputStreamReader = null;
        OutputStreamWriter outputStreamWriter = null;
        BufferedReader bufferedReader = null;
        BufferedWriter bufferedWriter = null;


        ServerSocket serverSocket = new ServerSocket(8080);


        while(true){
            try{
                socket = serverSocket.accept();
                inputStreamReader = new InputStreamReader(socket.getInputStream());// recupere le flux venant du serveur
                outputStreamWriter = new OutputStreamWriter(socket.getOutputStream());//flux pour envoyer au serveur
                bufferedReader = new BufferedReader(inputStreamReader);// convertit les bytes en texte
                bufferedWriter = new BufferedWriter(outputStreamWriter);// convertit le texte en bytes


                while (true){
                    String msgfromClient = bufferedReader.readLine();

                    System.out.println("Client : " + msgfromClient);

                    bufferedWriter.write("MSG Received");
                    bufferedWriter.newLine();
                    bufferedWriter.flush();

                    if (msgfromClient.equalsIgnoreCase("OK BYE")){
                        break ;
                    }
                }

            } catch (IOException e){
                e.printStackTrace();
            } finally {
                try{
                    if (serverSocket != null){serverSocket.close();}
                }catch (IOException e ){e.printStackTrace();}
            }
        }
    }
}
```

This time, we use a Socket instead of DatagramSocket because we are working with TCP, not UDP, which ensures a reliable and ordered flow of data between the client and server. The InputStreamReader reads the incoming byte stream from the server and converts it into readable characters, while the OutputStreamWriter transforms the client's characters into bytes to send. BufferedReader and BufferedWriter add buffering, allowing us to read and write line by line efficiently, making it easier to send complete messages through the TCP stream

Tristan BRAU
Léo PHAV

2.2 TCP Client

```java
public class TCP_Client{


    Run | Debug
    public static void main(String[] args){

        Socket socket = null;
        InputStreamReader inputStreamReader = null;
        OutputStreamWriter outputStreamWriter = null;
        BufferedReader bufferedReader = null;
        BufferedWriter bufferedWriter = null;

        String host = args[0];
        int port = Integer.parseInt(args[1]);
        try{
            socket = new Socket(host, port);
            inputStreamReader = new InputStreamReader(socket.getInputStream());// recupere le flux venant du serveur
            outputStreamWriter = new OutputStreamWriter(socket.getOutputStream());//flux pour envoyer au serveur
            bufferedReader = new BufferedReader(inputStreamReader);// convertit les bytes en texte
            bufferedWriter = new BufferedWriter(outputStreamWriter);// convertit le texte en bytes
            Scanner scanner = new Scanner(System.in);// permet d elire ce que tu lis dans la console

            while (true){
                String msgtoSend = scanner.nextLine();

                bufferedWriter.write(msgtoSend);
                bufferedWriter.newLine();
                bufferedWriter.flush();

                System.out.println("Server : " + bufferedReader.readLine());
                if (msgtoSend.equalsIgnoreCase("BYE")){
                    break;
                }


            }
        }catch (IOException e) {
                e.printStackTrace();
        }finally{
            try{
                if (socket != null){socket.close();}
                if (bufferedReader != null) bufferedReader.close();
                if (bufferedWriter != null) bufferedWriter.close();
                if (inputStreamReader != null) inputStreamReader.close();
                if (outputStreamWriter != null) outputStreamWriter.close();
            } catch (IOException e ){
                e.printStackTrace();
            }
```

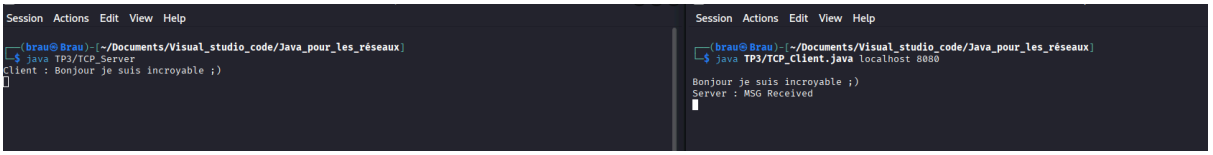
In the client part, the program reads input from the user using a Scanner, sends it to the server through the BufferedWriter, and waits for a response using BufferedReader.readLine(). In contrast, the server part listens for messages from the client using BufferedReader.readLine(), prints the received message, and sends an acknowledgment back with BufferedWriter. Both sides check for specific keywords like "BYE" to terminate the connection. This separation ensures that the client initiates communication while the server responds and handles incoming messages.

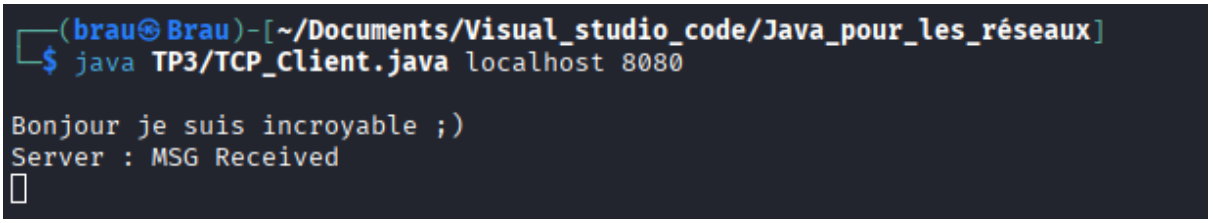
Tristan BRAU
Léo PHAV

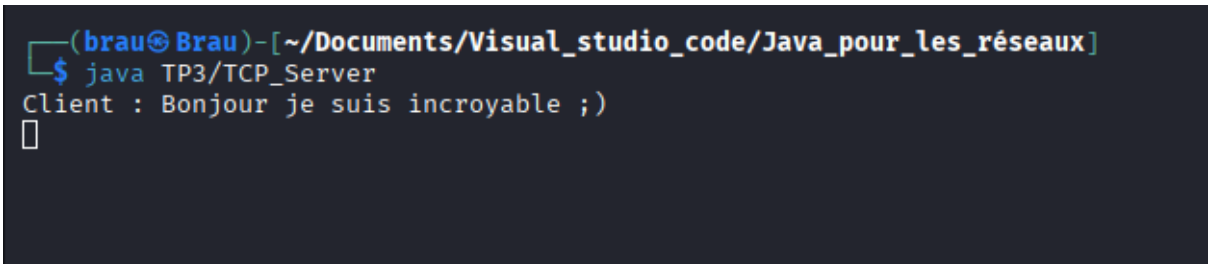
Results :

At the end, the TCP client works similarly to a UDP client in terms of starting the program from the command line. The command:

$ java TCPClient localhost 8080







If the client types the word "BYE", the session will close and the connection with the server will terminate.

Here, we can see that numbers and special characters can be transmitted as well: