

Java pour les réseaux - TP 2

1.1 UDP Server

Tristan BRAU Part:

```
package TP2;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.SocketException;
import java.io.IOException;
import java.net DatagramPacket;

public class UDP_Server{
    private DatagramSocket datagramSocket;// Doit tj être ouvert pour recevoir ou envoyer contrairement à datagrampacket qui n'a pas besoin d'être ouvert
    private byte[] buffer = new byte[256];

    public UDP_Server(DatagramSocket datagramSocket){
        this.datagramSocket = datagramSocket;
    }

    public void receiveThenSend(){
        while (true){
            try{
                DatagramPacket datagramPacket = new DatagramPacket(buffer, buffer.length);
                datagramSocket.receive(datagramPacket);
                InetAddress inetAddress = datagramPacket.getAddress(); //récupérer l'adresse IP de l'expéditeur du paquet UDP reçu.
                int port = datagramPacket.getPort();
                String messagefromClient = new String(datagramPacket.getData(),datagramPacket.getOffset(),datagramPacket.getLength());
                System.out.println("Message from client : " + messagefromClient + "\n");
                datagramPacket = new DatagramPacket(buffer, buffer.length, inetAddress, port);
                datagramSocket.send(datagramPacket);
            }catch (IOException e) {
                e.printStackTrace();
                break;
            }
        }
    }

    public static void main(String[] args) throws SocketException{
        DatagramSocket datagramSocket = new DatagramSocket(8080); // port = 8080
        UDP_Server server = new UDP_Server(datagramSocket);
        server.receiveThenSend();
    }
}
```

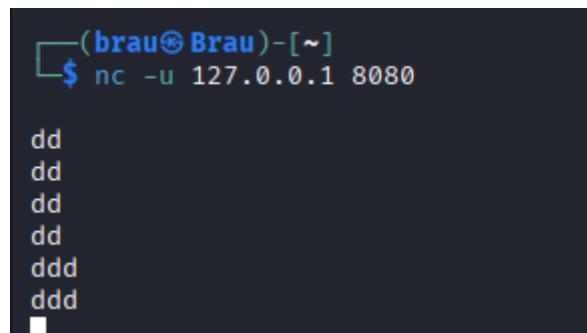
To test this server, you can first use the netcat command (nc), from another terminal.

```
[brau@Brau] ~/Documents/Visual_studio_code/Java_pour_les_réseaux]
$ java TP2/UDP_Server

Message from client : dd

Message from client : dd

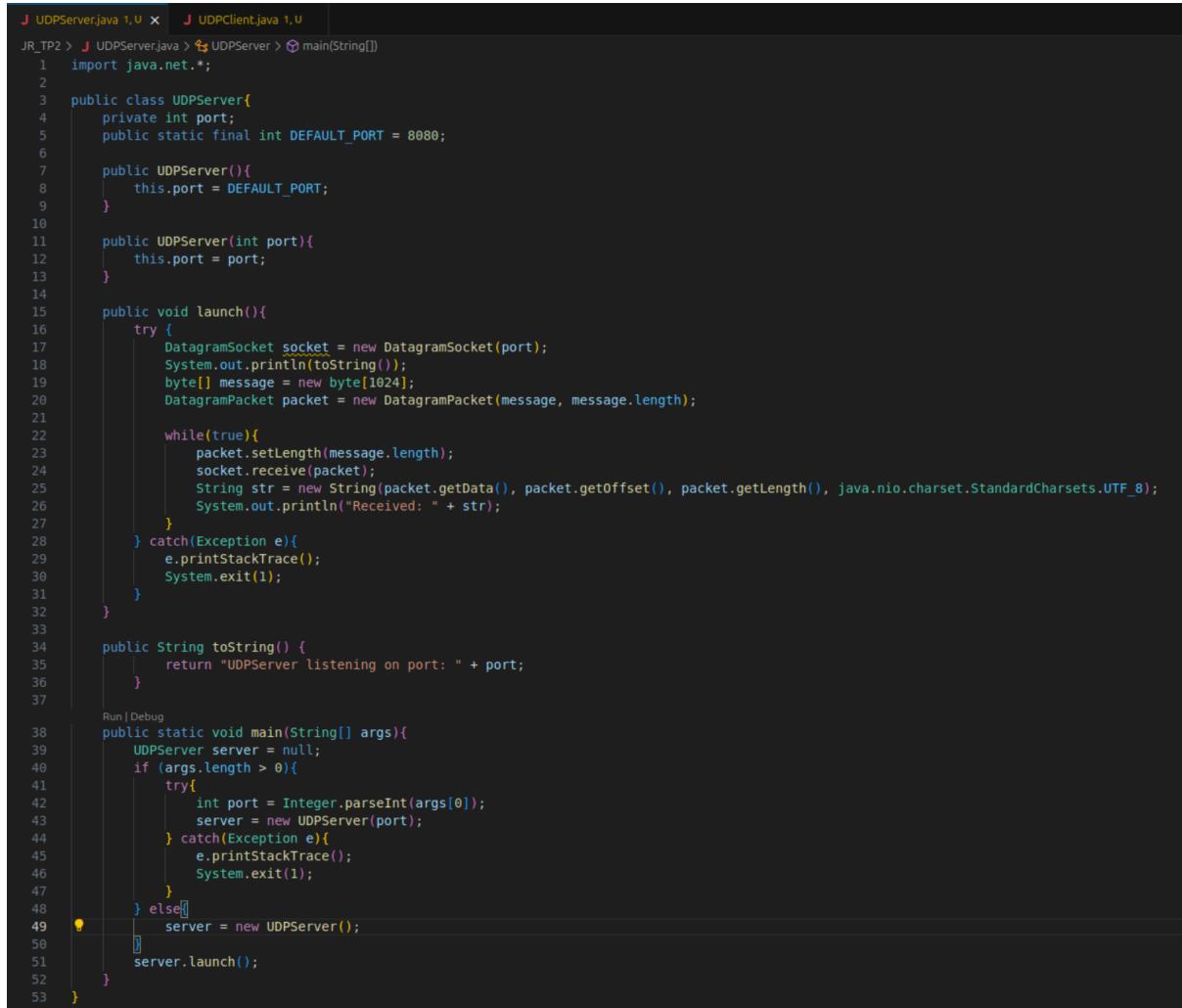
Message from client : ddd
```



```
(brau@Brau)~]$ nc -u 127.0.0.1 8080
dd
dd
dd
dd
ddd
ddd
```

Léo PHAV Part:

The program **UDPServer.java** creates a simple UDP server that listens for and displays messages sent by a client. When launched, it opens a UDP socket on a specific port, either the default port 8080 or one provided as an argument. The server continuously waits for incoming datagram packets, decodes the received data in UTF-8, and prints the message to the console.



```
J UDPServer.java 1,0 x J UDPClient.java 1,0
JR_TP2 > J UDPServer.java > UDPServer > main(String[])
1 import java.net.*;
2
3 public class UDPServer{
4     private int port;
5     public static final int DEFAULT_PORT = 8080;
6
7     public UDPServer(){
8         this.port = DEFAULT_PORT;
9     }
10
11    public UDPServer(int port){
12        this.port = port;
13    }
14
15    public void launch(){
16        try {
17            DatagramSocket socket = new DatagramSocket(port);
18            System.out.println(toString());
19            byte[] message = new byte[1024];
20            DatagramPacket packet = new DatagramPacket(message, message.length);
21
22            while(true){
23                packet.setLength(message.length);
24                socket.receive(packet);
25                String str = new String(packet.getData(), packet.getOffset(), packet.getLength(), java.nio.charset.StandardCharsets.UTF_8);
26                System.out.println("Received: " + str);
27            }
28        } catch(Exception e){
29            e.printStackTrace();
30            System.exit(1);
31        }
32    }
33
34    public String toString() {
35        return "UDPServer listening on port: " + port;
36    }
37
Run|Debug
38    public static void main(String[] args){
39        UDPServer server = null;
40        if (args.length > 0){
41            try{
42                int port = Integer.parseInt(args[0]);
43                server = new UDPServer(port);
44            } catch(Exception e){
45                e.printStackTrace();
46                System.exit(1);
47            }
48        } else{
49            server = new UDPServer();
50        }
51        server.launch();
52    }
53 }
```

Once I finished the code, I launched the server and tried to send informations with the command: `nc -u localhost [port]`

First, I tried with the default port (8080), the server received correctly the messages:

```
leo-phav@leo-phav:~/Dev/VSCode/JR_TP2$ java UDPServer
UDPServer listening on port: 8080
Received: leo
Received: RTS
Received: 3FISE
```

```
leo-phav@leo-phav:~/Dev/VSCode/JR_TP2$ nc -u localhost 8080
leo
RTS
3FISE
```

Then, with port 9090:

```
leo-phav@leo-phav:~/Dev/VSCode/JR_TP2$ java UDPServer 9090
UDPServer listening on port: 9090
Received: Hello
Received: 123456
```

```
leo-phav@leo-phav:~/Dev/VSCode/JR_TP2$ nc -u localhost 9090
Hello
123456
```

1.2 UDP Client

Tristan BRAU Part:

```
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.SocketException;
import java.util.Scanner;
import java.net.UnknownHostException;

public class UDP_Client {
    private DatagramSocket datagramSocket;
    private InetAddress inetAdress;
    private byte[] buffer ;

    public UDP_Client(DatagramSocket datagramSocket,InetAddress inetAdress){
        this.datagramSocket = datagramSocket;
        this.inetAdress = inetAdress;
    }

    public void sendThenReceive(){
        Scanner scanner = new Scanner(System.in);
        while(true){
            try{
                String messageToSend = scanner.nextLine();
                buffer = messageToSend.getBytes();
                DatagramPacket datagramPacket = new DatagramPacket(buffer, buffer.length,inetAdress,8080);
                datagramSocket.send(datagramPacket);
                datagramSocket.receive(datagramPacket);
                String messageFromServer = new String(datagramPacket.getData(),0,datagramPacket.getLength());
                System.out.println("The server say you say" + messageFromServer);
            } catch(IOException e){
                e.printStackTrace();
                break;
            }
        }
    }

    Run|Debug
    public static void main(String[] args) throws SocketException, UnknownHostException{
        DatagramSocket datagramSocket = new DatagramSocket();
        InetAddress inetAddress = InetAddress.getByName("localhost");
        UDP_Client client = new UDP_Client(datagramSocket,inetAddress);
        System.out.println("Send datagram packets to a server UDP ");
        client.sendThenReceive();
    }
}
```

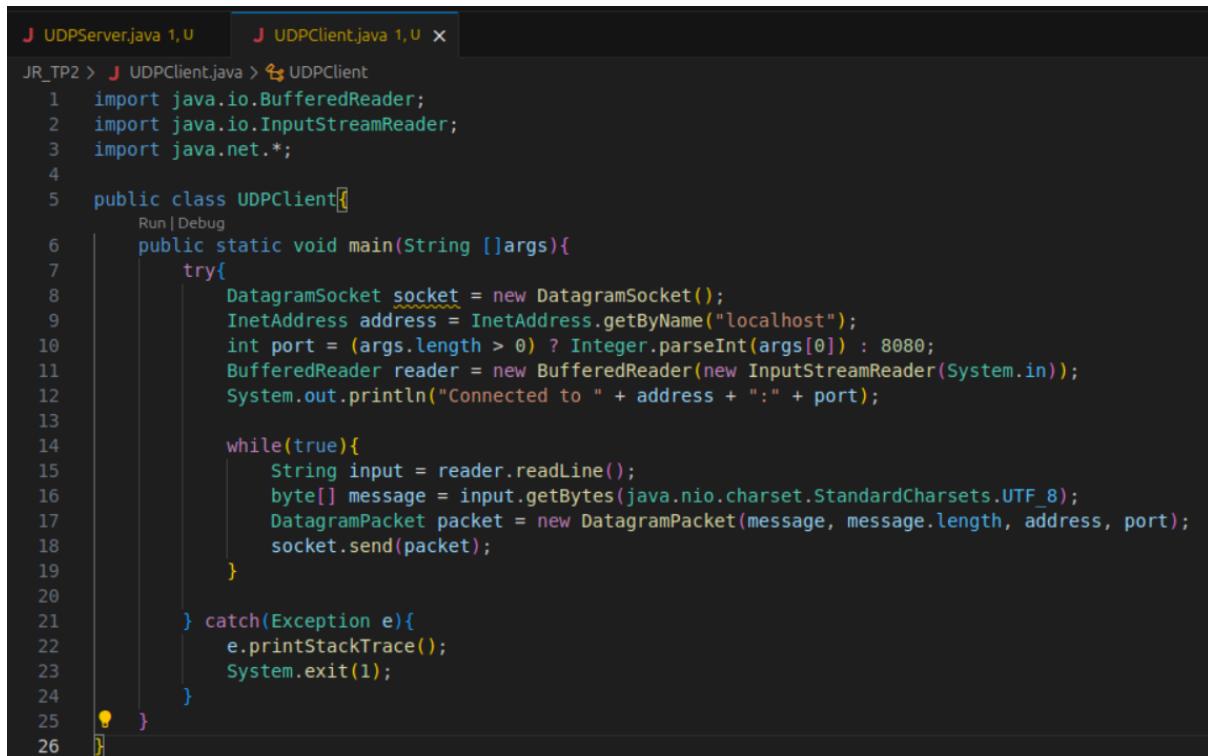
```
[brau@Brau] - [~/Documents/Visual_studio_code/Java_pour_les_réseaux]
$ /usr/bin/env /usr/lib/jvm/java-21-openjdk-amd64/bin/java -XX:+ShowCodeDetail
les_réseaux_c7927af9/bin TP2.UDP_Server
Message from client : Hi How are you server today
```

```
[brau@Brau] - [~/Documents/Visual_studio_code/Java_pour_les_reseaux]
$ /usr/bin/env /usr/lib/jvm/java-21-openjdk-amd64/bin/java -XX:+ShowCodeDetail
les_reseaux_c7927af9/bin TP2.UDP_Client
Send datagram packets to a server UDP
Hi How are you server today
The server say you sayHi How are you server today
[]
```

We successfully receive the messages sent from the client on our UDP server, and the server correctly sends them back to the client.

Léo PHAV Part:

The program **UDPClient.java** implements a simple client that sends messages to a UDP server. When executed, it creates a UDP socket and connects to a server located at localhost on a specific port — either the default port 8080 or a port provided as a command-line argument. The program then reads user input from the keyboard in a continuous loop. Each line entered is converted into bytes encoded in UTF-8 and sent to the server using a DatagramPacket.



A screenshot of the VS Code interface showing the `UDPClient.java` file. The code implements a UDP client with a main method that creates a DatagramSocket, reads input from the user, and sends it to a localhost server on port 8080 or a specified port. It handles exceptions and exits if one occurs.

```
1 import java.io.BufferedReader;
2 import java.io.InputStreamReader;
3 import java.net.*;
4
5 public class UDPClient{
6     public static void main(String []args){
7         try{
8             DatagramSocket socket = new DatagramSocket();
9             InetAddress address = InetAddress.getByName("localhost");
10            int port = (args.length > 0) ? Integer.parseInt(args[0]) : 8080;
11            BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
12            System.out.println("Connected to " + address + ":" + port);
13
14            while(true){
15                String input = reader.readLine();
16                byte[] message = input.getBytes(java.nio.charset.StandardCharsets.UTF_8);
17                DatagramPacket packet = new DatagramPacket(message, message.length, address, port);
18                socket.send(packet);
19            }
20        } catch(Exception e){
21            e.printStackTrace();
22            System.exit(1);
23        }
24    }
25 }
26 }
```

Here, I did the same manipulation but with the UDP Client:



A terminal session showing the interaction between a UDP Server and a UDP Client. The server (left) listens on port 8080 and receives messages from the client (right) on port 127.0.0.1:8080. The client sends messages to the server, which then prints them out.

```
leo-phav@leo-phav:~/Dev/VSCode/JR_TP2$ java UDPServer
UDPServer listening on port: 8080
Received: Léo
Received: Tristan
Received: RTS3
[blank]
leo-phav@leo-phav:~/Dev/VSCode/JR_TP2$ java UDPClient
Connected to localhost/127.0.0.1:8080
Léo
Tristan
RTS3
[blank]

leo-phav@leo-phav:~/Dev/VSCode/JR_TP2$ java UDPServer 9090
UDPServer listening on port: 9090
Received: Hello from UDPServer
Received: Java Lab 2
[blank]
leo-phav@leo-phav:~/Dev/VSCode/JR_TP2$ java UDPClient 9090
Connected to localhost/127.0.0.1:9090
Hello from UDPServer
Java Lab 2
[blank]
```

All messages have been received with no errors.

1.3 Additional Parts

Tristan BRAU Part:

Add sequence numbers to messages

```
[brau@Brau] - [/Documents/Visual_studio_code/Java_pour_les_réseaux]
$ /usr/bin/env /usr/lib/jvm/java-21-openjdk-amd64/bin/java -XX:+ShowCodeDetailsInExceptionMessage TP2.UDP_Server
les_réseaux_c7927af9/bin TP2.UDP_Server
Message from client : Seq = 0;ok mec

Message from client : Seq = 0 | ok mec

Message from client : Seq = 1 | ok mec

Message from client : Seq = 2 |
```

```
[brau@Brau] - [/Documents/Visual_studio_code/Java_pour_les_réseaux]
$ /usr/bin/env /usr/lib/jvm/java-21-openjdk-amd64/bin/java -XX:+ShowCodeDetailsInExceptionMessage TP2.UDP_Client
les_réseaux_c7927af9/bin TP2.UDP_Client
Send datagram packets to a server UDP
ok mec
Seq = 0 | The server say you say Seq = 0 | ok mec
ok mec
Seq = 1 | The server say you say Seq = 1 | ok mec
Seq = 2 | The server say you say Seq = 2 |
```

Léo PHAV Part:

To implement a sequence number, I created a new variable:

```
String seqMessage = sequenceNum + " | " + input;
```

I added a sequence number into my messages but when I sent empty messages, the sequence number still increment:

```
leo-phav@leo-phav:~/Dev/VSCode/JR_TP2$ java UDPServer
UDPServer listening on port: 8080
Received: 1 | Hi
Received: 2 | Léo
Received: 3 | RTS
Received: 4 |
Received: 5 |
Received: 6 |
Received: 7 |
Received: 8 | 12345
leo-phav@leo-phav:~/Dev/VSCode/JR_TP2$ java UDPClient
Connected to localhost/127.0.0.1:8080
Hi
Léo
RTS
12345
```

By adding this command before I create seqMessage variable, I solve the issue:

```
if (input == null || input.isEmpty()) continue;
```

```
leo-phav@leo-phav:~/Dev/VSCode/JR_TP2$ java UDPServer
UDPServer listening on port: 8080
Received: 1 | Léo
Received: 2 | PHAV
Received: 3 | UDP
Received: 4 | RTS
Received: 5 | 12345
[]

leo-phav@leo-phav:~/Dev/VSCode/JR_TP2$ java UDPClient
Connected to localhost/127.0.0.1:8080
Léo
PHAV
UDP

RTS
12345
[]
```