An-Najah National University

Department of Computer Engineering

Networks 1 (10636454)

First Semester 2021/2022

جامعة النجاح الوطنية

قسم هندسة الحاسوب

شبكات الحاسوب 1 10636454

الفصل الأول 2022/2021

Instructor : د . عبدالله راشد

NAME	IDNUM	SEC-2
Ahmad Abdullah AL-Qerem	11819195	12:30 – 02:00

Programming Assignment

Introduction :

In this assignment I had implemented simple Socket in java code that organize the connection between client and server , in two approach :

1- UDP
2- TCP

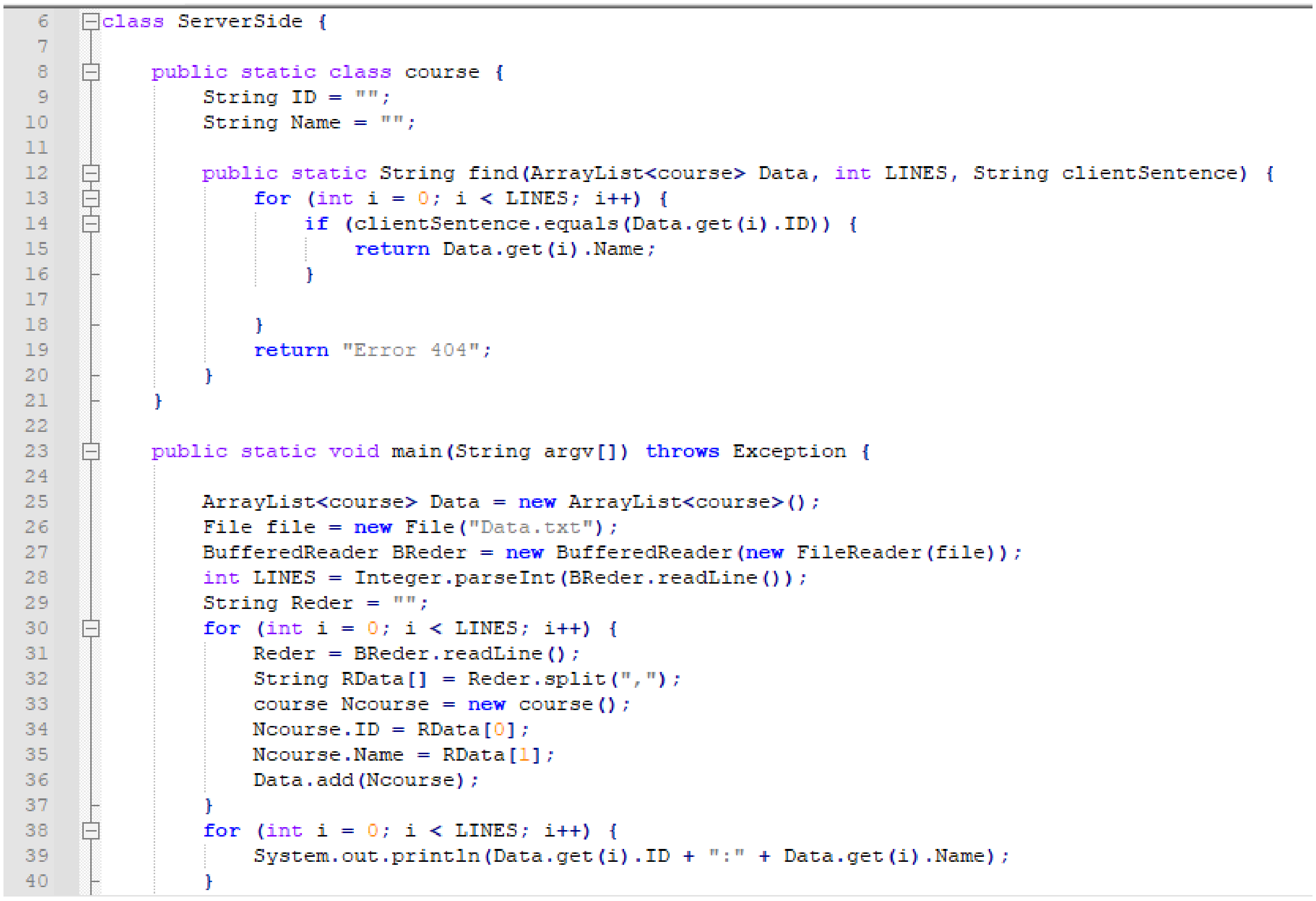
The server side (response) is taken from a file stored in the server side (assumed a Data Base table).For example: the client sends 10636454, the server sends Networks1. In case there is no match request, send Error 404.

Solution :

Part 1 (TCP):

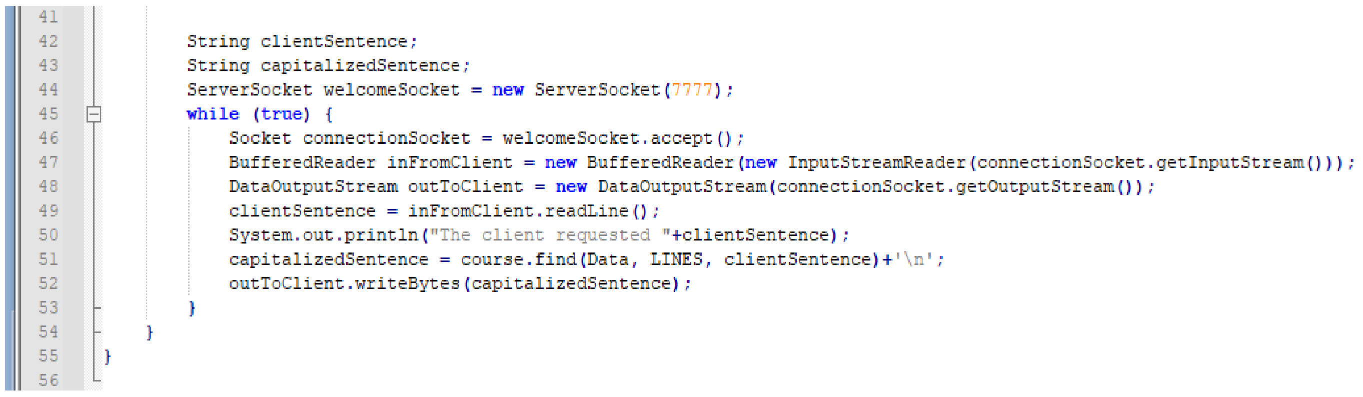
Server Side

I had written the following code :



```
class ServerSide {

    public static class course {
        String ID = "";
        String Name = "";

        public static String find(ArrayList<course> Data, int LINES, String clientSentence) {
            for (int i = 0; i < LINES; i++) {
                if (clientSentence.equals(Data.get(i).ID)) {
                    return Data.get(i).Name;
                }

            }
            return "Error 404";
        }
    }

    public static void main(String argv[]) throws Exception {

        ArrayList<course> Data = new ArrayList<course>();
        File file = new File("Data.txt");
        BufferedReader BReder = new BufferedReader(new FileReader(file));
        int LINES = Integer.parseInt(BReder.readLine());
        String Reder = "";
        for (int i = 0; i < LINES; i++) {
            Reder = BReder.readLine();
            String RData[] = Reder.split(",");
            course Ncourse = new course();
            Ncourse.ID = RData[0];
            Ncourse.Name = RData[1];
            Data.add(Ncourse);
        }
        for (int i = 0; i < LINES; i++) {
            System.out.println(Data.get(i).ID + ":" + Data.get(i).Name);
        }
```

At first I created a buffered to read lines from file such that the first line contain the number of lines in the file, in for loop continue reading lines and store the data in ArrayList of type course to search in .then the server will print all data stored .

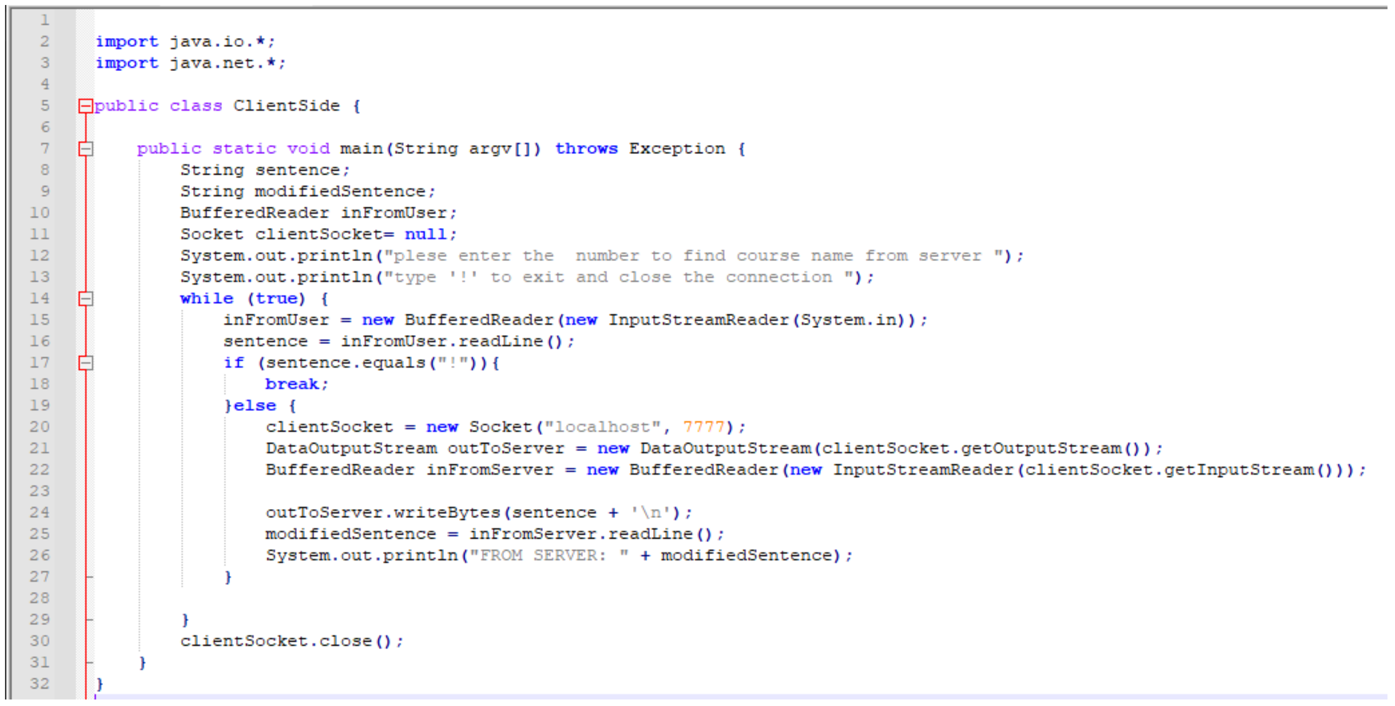


```java

        String clientSentence;
        String capitalizedSentence;
        ServerSocket welcomeSocket = new ServerSocket(7777);
        while (true) {
            Socket connectionSocket = welcomeSocket.accept();
            BufferedReader inFromClient = new BufferedReader(new InputStreamReader(connectionSocket.getInputStream()));
            DataOutputStream outToClient = new DataOutputStream(connectionSocket.getOutputStream());
            clientSentence = inFromClient.readLine();
            System.out.println("The client requested "+clientSentence);
            capitalizedSentence = course.find(Data, LINES, clientSentence)+'\n';
            outToClient.writeBytes(capitalizedSentence);
        }
    }
}

```

- Here is the server code , at first I created a server socket at port (“ 7777 ”) , then inter the while loop (the server is listening).
- Create a socket to connect it with the client .
- Create a buffer to read data that arrived from client .
- Then the static function (“ find (   ) ”) will return the name of course depends on course number received if the server didn’t find the name it will return (“ Error 404 ”).

Client Side

I had written the following code for client :



```java

import java.io.*;
import java.net.*;

public class ClientSide {

    public static void main(String argv[]) throws Exception {
        String sentence;
        String modifiedSentence;
        BufferedReader inFromUser;
        Socket clientSocket= null;
        System.out.println("plese enter the  number to find course name from server ");
        System.out.println("type '!' to exit and close the connection ");
        while (true) {
            inFromUser = new BufferedReader(new InputStreamReader(System.in));
            sentence = inFromUser.readLine();
            if (sentence.equals("!")){
                break;
            }else {
                clientSocket = new Socket("localhost", 7777);
                DataOutputStream outToServer = new DataOutputStream(clientSocket.getOutputStream());
                BufferedReader inFromServer = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

                outToServer.writeBytes(sentence + '\n');
                modifiedSentence = inFromServer.readLine();
                System.out.println("FROM SERVER: " + modifiedSentence);
            }

        }
        clientSocket.close();
    }
}
```

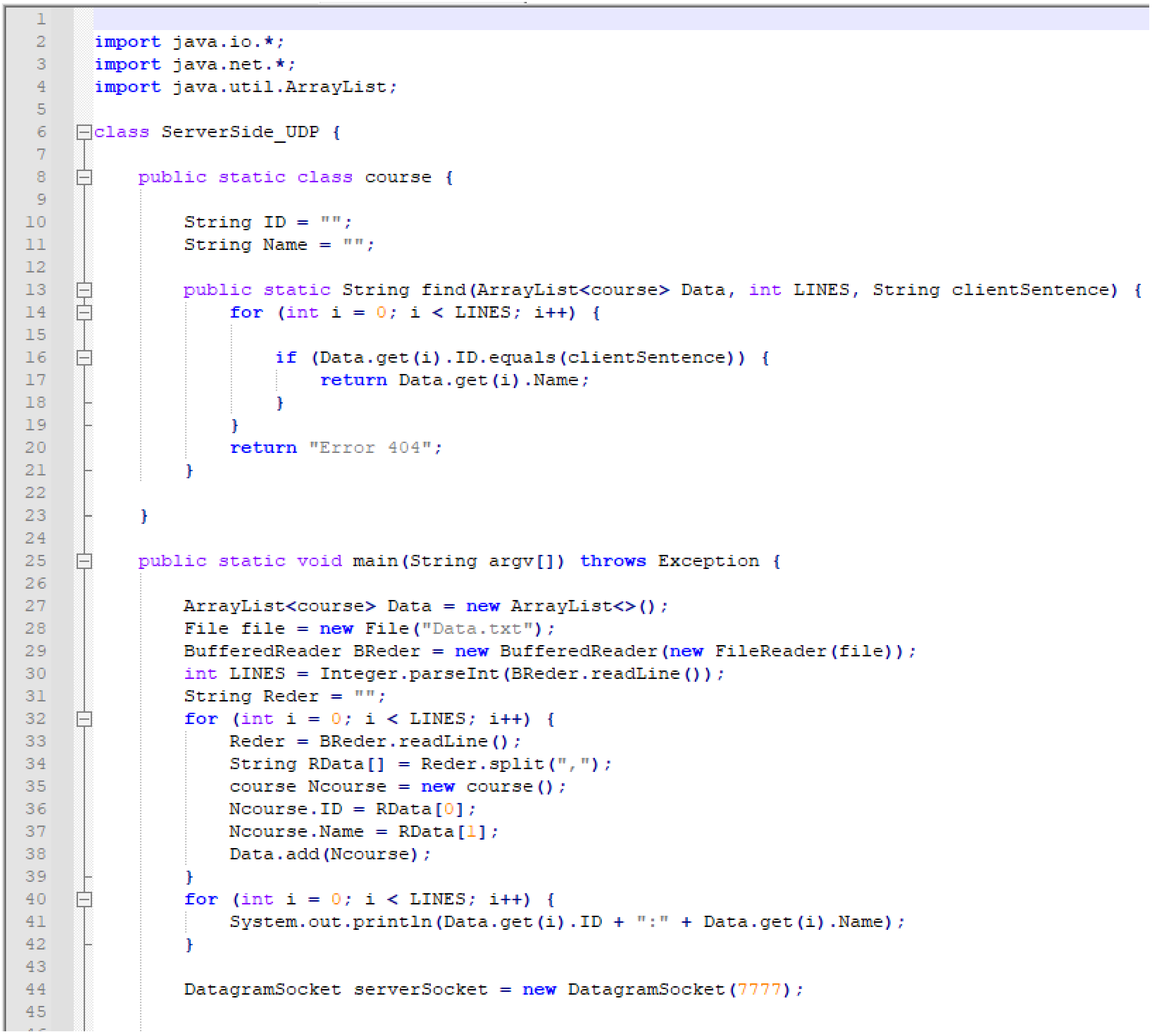
- The same as server it will create new socket and prepare it to communicate with server .
- While loop here used to achieve assignment requirements (“ The client can make more requests, before closing the app ”) , the client can close connection if he typed (“ ! ”).

- Each time in while loop read a line from client consol and check it if it’s not (“ ! ”) then send it to server at port (“ 7777 ”) .
- Else will close the connection .

Part 2 (UDP):

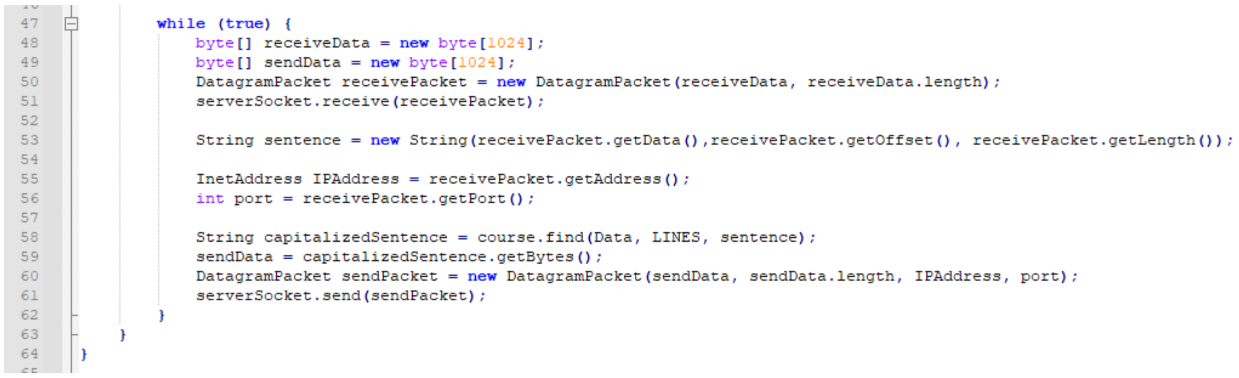
Server Side

I had written the following code :



```java

import java.io.*;
import java.net.*;
import java.util.ArrayList;

class ServerSide_UDP {

    public static class course {

        String ID = "";
        String Name = "";

        public static String find(ArrayList<course> Data, int LINES, String clientSentence) {
            for (int i = 0; i < LINES; i++) {

                if (Data.get(i).ID.equals(clientSentence)) {
                    return Data.get(i).Name;
                }
            }
            return "Error 404";
        }

    }

    public static void main(String argv[]) throws Exception {

        ArrayList<course> Data = new ArrayList<>();
        File file = new File("Data.txt");
        BufferedReader BReder = new BufferedReader(new FileReader(file));
        int LINES = Integer.parseInt(BReder.readLine());
        String Reder = "";
        for (int i = 0; i < LINES; i++) {
            Reder = BReder.readLine();
            String RData[] = Reder.split(",");
            course Ncourse = new course();
            Ncourse.ID = RData[0];
            Ncourse.Name = RData[1];
            Data.add(Ncourse);
        }
        for (int i = 0; i < LINES; i++) {
            System.out.println(Data.get(i).ID + ":" + Data.get(i).Name);
        }

        DatagramSocket serverSocket = new DatagramSocket(7777);

```

The same as TCP server **code …**

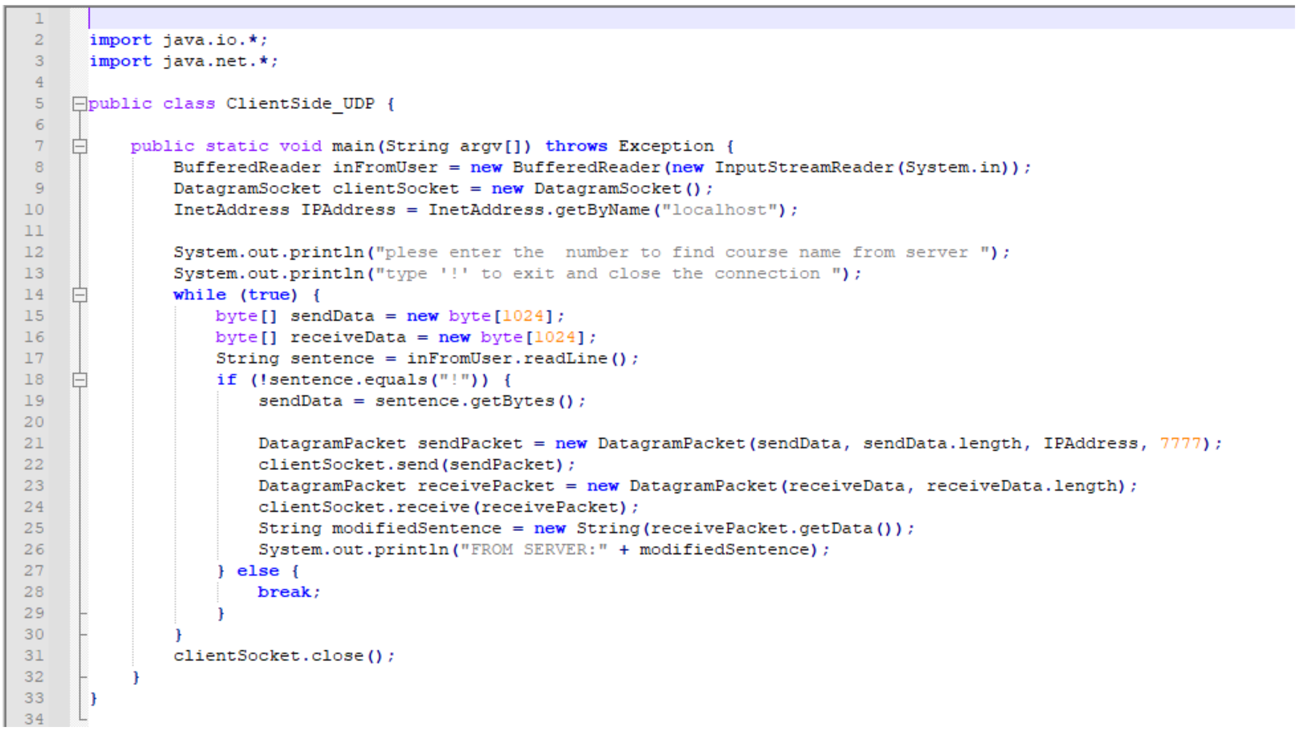


```java
        while (true) {
            byte[] receiveData = new byte[1024];
            byte[] sendData = new byte[1024];
            DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
            serverSocket.receive(receivePacket);

            String sentence = new String(receivePacket.getData(),receivePacket.getOffset(), receivePacket.getLength());

            InetAddress IPAddress = receivePacket.getAddress();
            int port = receivePacket.getPort();

            String capitalizedSentence = course.find(Data, LINES, sentence);
            sendData = capitalizedSentence.getBytes();
            DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, IPAddress, port);
            serverSocket.send(sendPacket);
        }
    }
}
```

But instead of creating a socket for each connection here the server will receive all packets in port (“ 7777 ”) with length of (“ 1024 byte ”).

Client Side

I had written the following code :



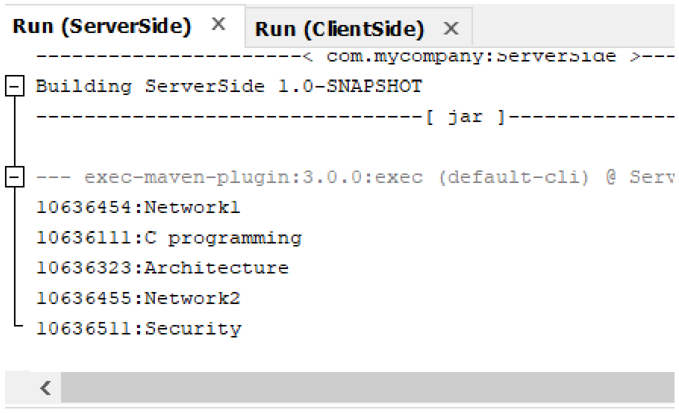
```java

import java.io.*;
import java.net.*;

public class ClientSide_UDP {

    public static void main(String argv[]) throws Exception {
        BufferedReader inFromUser = new BufferedReader(new InputStreamReader(System.in));
        DatagramSocket clientSocket = new DatagramSocket();
        InetAddress IPAddress = InetAddress.getByName("localhost");

        System.out.println("plese enter the  number to find course name from server ");
        System.out.println("type '!' to exit and close the connection ");
        while (true) {
            byte[] sendData = new byte[1024];
            byte[] receiveData = new byte[1024];
            String sentence = inFromUser.readLine();
            if (!sentence.equals("!")) {
                sendData = sentence.getBytes();

                DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, IPAddress, 7777);
                clientSocket.send(sendPacket);
                DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
                clientSocket.receive(receivePacket);
                String modifiedSentence = new String(receivePacket.getData());
                System.out.println("FROM SERVER:" + modifiedSentence);
            } else {
                break;
            }
        }
        clientSocket.close();
    }
}

```

The same as TCP client **code …**

But it will send data as packets .

Output: (TCP)

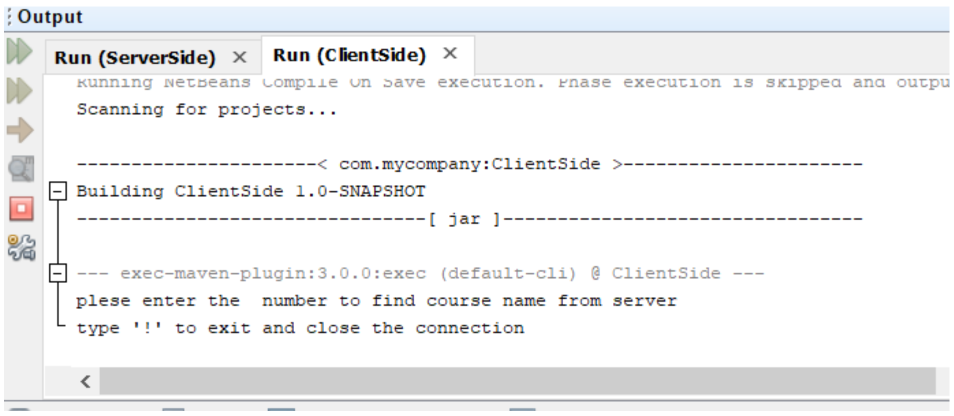
1- Running the server :



Run (ServerSide) ✕ Run (ClientSide) ✕

```
----------------------< com.mycompany:ServerSide >---
Building ServerSide 1.0-SNAPSHOT
--------------------------------[ jar ]--------------

--- exec-maven-plugin:3.0.0:exec (default-cli) @ Serv
10636454:Network1
10636111:C programming
10636323:Architecture
10636455:Network2
10636511:Security
```

2- Running the client :

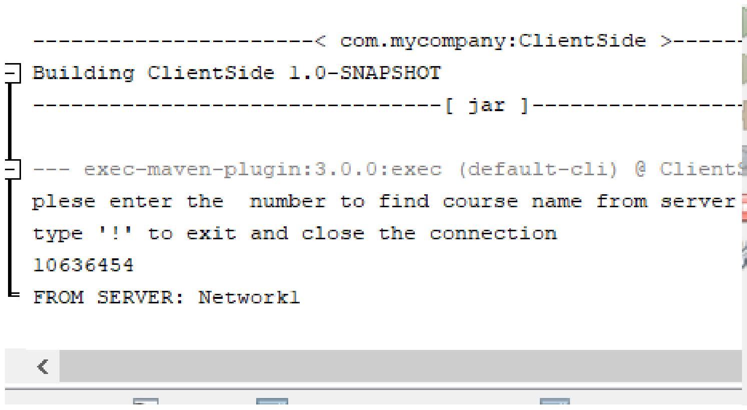


Output

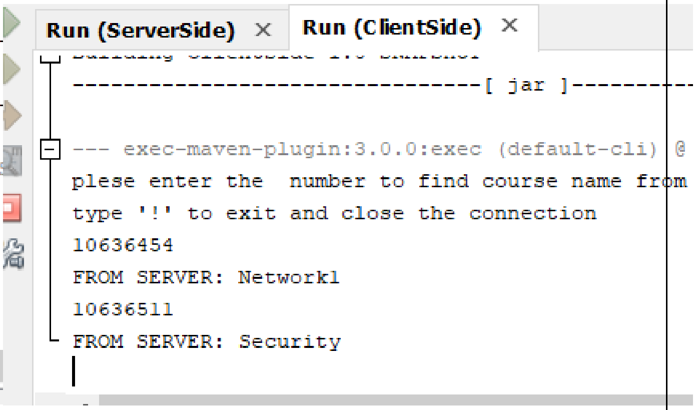
Run (ServerSide) ✕ Run (ClientSide) ✕

```
Running NetBeans Compile On Save execution. Phase execution is skipped and outpu
Scanning for projects...

----------------------< com.mycompany:ClientSide >----------------------
Building ClientSide 1.0-SNAPSHOT
--------------------------------[ jar ]---------------------------------

--- exec-maven-plugin:3.0.0:exec (default-cli) @ ClientSide ---
plese enter the  number to find course name from server
type '!' to exit and close the connection
```

3- Search at 10636454 which is Network1

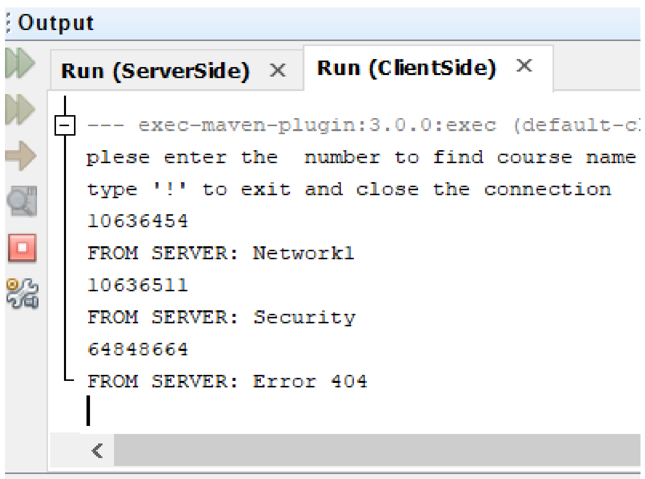


```
----------------------< com.mycompany:ClientSide >------
Building ClientSide 1.0-SNAPSHOT
--------------------------------[ jar ]-----------------

--- exec-maven-plugin:3.0.0:exec (default-cli) @ ClientS
plese enter the  number to find course name from server
type '!' to exit and close the connection
10636454
FROM SERVER: Network1
```



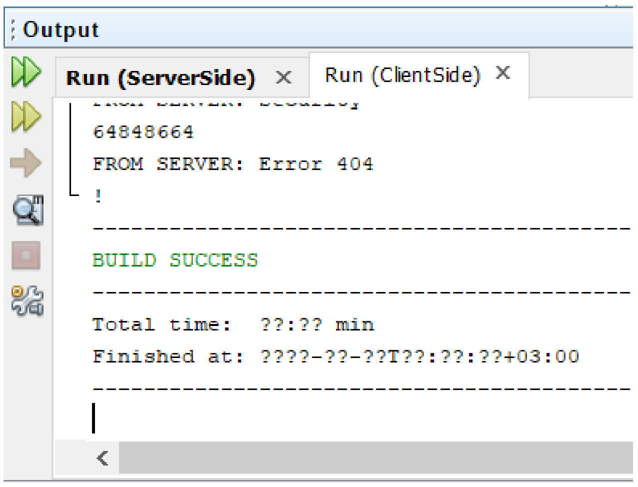
```
--------------------------------[ jar ]----------

--- exec-maven-plugin:3.0.0:exec (default-cli) @
plese enter the  number to find course name from
type '!' to exit and close the connection
10636454
FROM SERVER: Network1
10636511
FROM SERVER: Security
```

4- “ Error 404 “



```
--- exec-maven-plugin:3.0.0:exec (default-c
plese enter the  number to find course name
type '!' to exit and close the connection
10636454
FROM SERVER: Network1
10636511
FROM SERVER: Security
64848664
FROM SERVER: Error 404
```

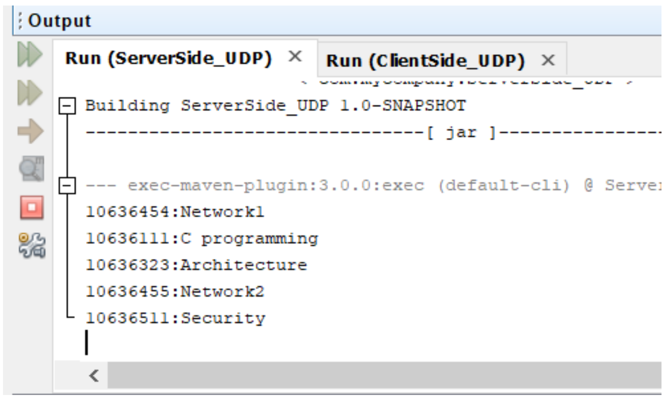
5- close connection (“ ! ”):



```
64848664
FROM SERVER: Error 404
!
------------------------------------------
BUILD SUCCESS
------------------------------------------
Total time:  ??:?? min
Finished at: ????-??-??T??:??:??+03:00
------------------------------------------
```

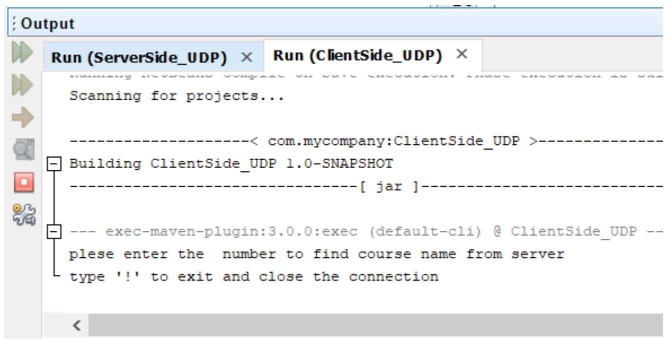
Output: (UDP)

1- Running the server :



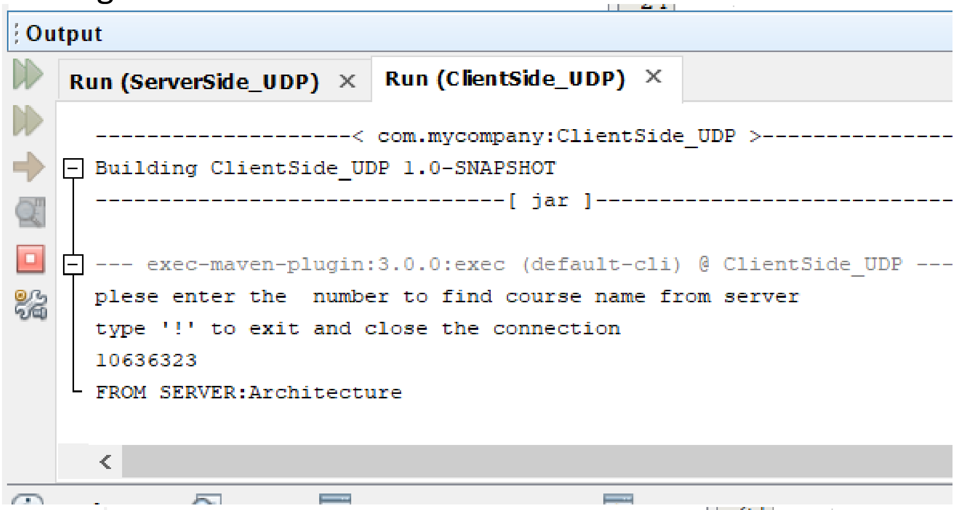
```
Building ServerSide_UDP 1.0-SNAPSHOT
--------------------------------[ jar ]----------------

--- exec-maven-plugin:3.0.0:exec (default-cli) @ Serve
10636454:Network1
10636111:C programming
10636323:Architecture
10636455:Network2
10636511:Security
```

2- Running the client :

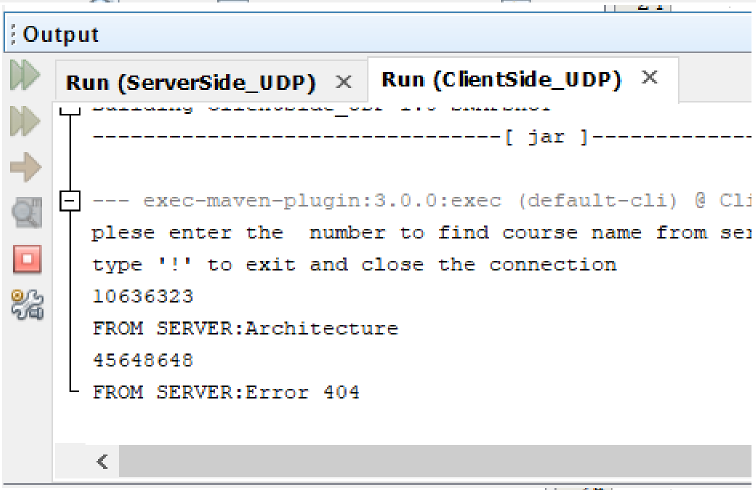


```
Scanning for projects...

-------------------< com.mycompany:ClientSide_UDP >--------------
Building ClientSide_UDP 1.0-SNAPSHOT
--------------------------------[ jar ]--------------------------

--- exec-maven-plugin:3.0.0:exec (default-cli) @ ClientSide_UDP --
plese enter the  number to find course name from server
type '!' to exit and close the connection
```

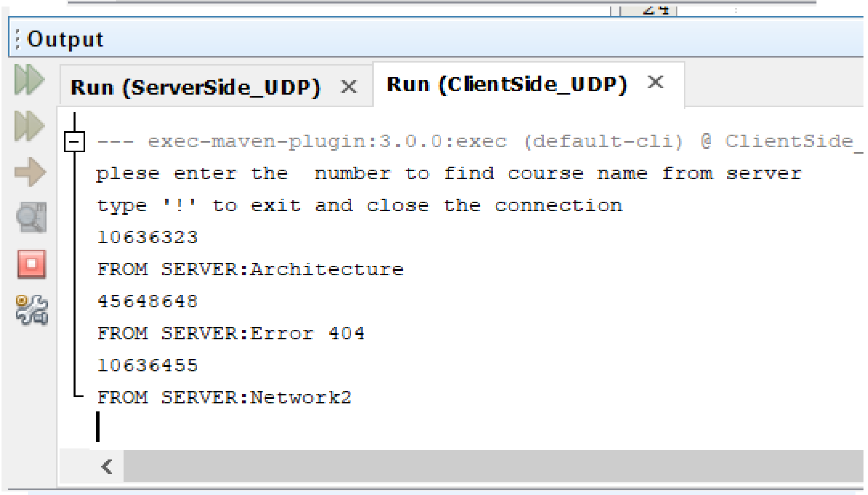
3- Testing :



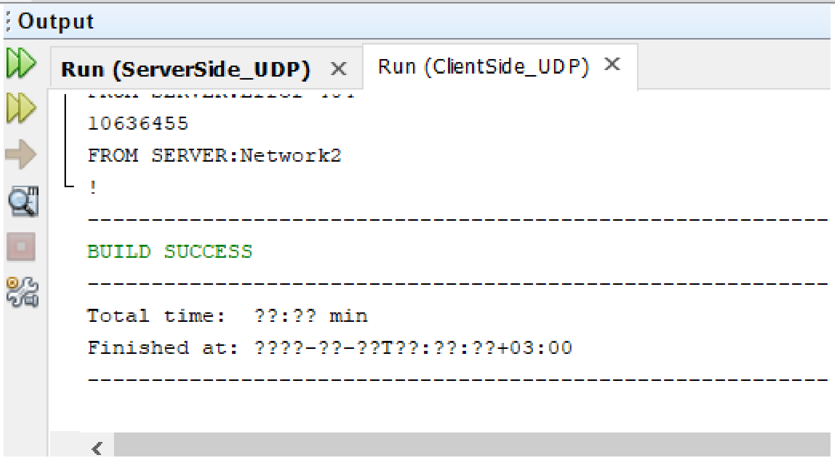
```
--------------------< com.mycompany:ClientSide_UDP >---------------
Building ClientSide_UDP 1.0-SNAPSHOT
--------------------------------[ jar ]----------------------------

--- exec-maven-plugin:3.0.0:exec (default-cli) @ ClientSide_UDP ---
plese enter the  number to find course name from server
type '!' to exit and close the connection
10636323
FROM SERVER:Architecture
```



```
--------------------------------[ jar ]------------

--- exec-maven-plugin:3.0.0:exec (default-cli) @ Cli
plese enter the  number to find course name from se
type '!' to exit and close the connection
10636323
FROM SERVER:Architecture
45648648
FROM SERVER:Error 404
```



```
--- exec-maven-plugin:3.0.0:exec (default-cli) @ ClientSide_
plese enter the  number to find course name from server
type '!' to exit and close the connection
10636323
FROM SERVER:Architecture
45648648
FROM SERVER:Error 404
10636455
FROM SERVER:Network2
```



```
10636455
FROM SERVER:Network2
!
------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------
Total time:  ??:?? min
Finished at: ????-??-??T??:??:??+03:00
------------------------------------------------------------
```