Exercise

Create a simplified FTP (file transport) client and server where the client can send or download text files from the server:

General use-cases

1.) Client connects to the server and sends a 'file listing' message
2.) Server sends back the list of the downloadable files
3.) Client lists the files and asks the user what action they want to take? Upload or download? ('u' or 'd')
4.) In both cases the uses must give the full file name with extension
5.) The client sends the selected file to the server (upload) or downloads the selected file from the server to a specific directory.

Server viewpoint

1.) After connecting, it reads the files from the /store subdirectory and sends the file names to the client after receiving the listing message.
2.) We are waiting for the client's 'u' or 'd' operation
3.) We get a filename from the client and if the action is 'd' (download), we read the file content and return its contents
4.) If the operation is 'u' (upload), we open a new file with the specified name and wait for the data to be written to the file.

Client viewpoint

1.) The client connects and waits for the list of files coming back and writes it to the console
2.) We ask for the “u” or “d” key
3.) Then we'll ask for the file-name as well.
4.) The client reads the files from the /files folder, or creates the downloaded file here
5.) If you press “d”, it creates /files/ and writes data from the server
6.) If you press “u”, /files/ is sent to the server

Alappéldák

1.) Hagyományos blokkolt TCP alapú socket szerver

Socket szerver kód

```
    import java.io.IOException;
    import java.io.ObjectInputStream;
```

```
    import java.io.ObjectOutputStream;
    import java.net.ServerSocket;
    import java.net.Socket;

    public class Server {
        ServerSocket providerSocket;
        Socket connection = null;
        ObjectOutputStream out;
        ObjectInputStream in;
        String message;

        Server() {
        }

        void run() {
            try {
                // 1. szerver socket létrehozása
                providerSocket = new ServerSocket(8080, 10);
                // 2. kapcsolódásra várakozás
                connection = providerSocket.accept();
                // 3. Input és Output streamek megadása
                out = new ObjectOutputStream(connection.getOutputStream());
                in = new ObjectInputStream(connection.getInputStream());
                // 4. socket kommunikáció
                do {
                    try {
                        message = (String) in.readObject();
                        System.out.println("client>" + message);
                        if (message.equals("bye")) {
                            sendMessage("bye");
                        }
                    } catch (ClassNotFoundException classnot) {
                        System.err.println("Data received in unknown
format");
                    }
                } while (!message.equals("bye"));
            } catch (IOException ioException) {
                ioException.printStackTrace();
            } finally {
                // 4: kapcsolat lezárása
                try {
                    in.close();
                    out.close();
                    providerSocket.close();
                } catch (IOException ioException) {
                    ioException.printStackTrace();
                }
            }
        }
```

```
        void sendMessage(String msg) {
            try {
                out.writeObject(msg);
                out.flush();
                System.out.println("server>" + msg);
            } catch (IOException ioException) {
                ioException.printStackTrace();
            }
        }

        public static void main(String args[]) {
            Server server = new Server();
            while (true) {
                server.run();
            }
        }
    }
```

Socket kliens kód

```
    import java.io.IOException;
    import java.io.ObjectInputStream;
    import java.io.ObjectOutputStream;
    import java.net.Socket;
    import java.net.UnknownHostException;

    public class Client {
        Socket requestSocket;
        ObjectOutputStream out;
        ObjectInputStream in;
        String message;

        Client() {
        }

        void run() {
            try {
                // 1. socket kapcsolat létrehozása
                requestSocket = new Socket("localhost", 8080);
                // 2. Input and Output streamek
                out = new
ObjectOutputStream(requestSocket.getOutputStream());
                in = new ObjectInputStream(requestSocket.getInputStream());
                // 3: Kommunikáció
                do {
                    try {
                        sendMessage("Hello szerver");
                        sendMessage("bye");
                        message = (String) in.readObject();
                    } catch (Exception e) {
```

```
                        System.err.println("data received in unknown
format");
                    }
                } while (!message.equals("bye"));
            } catch (UnknownHostException unknownHost) {
                System.err.println("You are trying to connect to an unknown
host!");
            } catch (IOException ioException) {
                ioException.printStackTrace();
            } finally {
                // 4: Kapcsolat zárása
                try {
                    in.close();
                    out.close();
                    requestSocket.close();
                } catch (IOException ioException) {
                    ioException.printStackTrace();
                }
            }
        }

        void sendMessage(String msg) {
            try {
                out.writeObject(msg);
                out.flush();
                System.out.println("client>" + msg);
            } catch (IOException ioException) {
                ioException.printStackTrace();
            }
        }

        public static void main(String args[]) {
            Client client = new Client();
            client.run();
        }
    }
```

2.) Hagyományos UDP alapú kommunikáció

2.a) Az alábbi Ágens küld egy üzenetet és a 8080-as porton várja a választ rá, ugyancsak UDP-vel. Az eclipse fejlesztőkörnyezetben a consolon beírt szöveget ctrl+z leütésével lehet elküldeni.

Feladat: módosítsuk a kódot, hogy át tudjon küldeni egy beégetett nevű, és létező, 2 kbyte-nál nagyobb szöveges vagy kép állományt és ellenőrizzük a sikeres küldést.

```
  package org.ait;

  import java.io.BufferedReader;
  import java.io.InputStreamReader;
```

```
  import java.net.DatagramPacket;
  import java.net.DatagramSocket;
  import java.net.InetAddress;

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

        byte[] sendData = new byte[1024];
        byte[] receiveData = new byte[1024];

        String sentence = inFromUser.readLine();
        sendData = sentence.getBytes();

        DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, IPAddress, 8080);
        clientSocket.send(sendPacket);

        DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
        clientSocket.receive(receivePacket);
        String modifiedSentence = new String(receivePacket.getData());

        System.out.println("átalakítva:" + modifiedSentence);
        clientSocket.close();
    }
    }
```

2.b) Az UDP szerver a 8080-as porton várja az ágensek üzeneteit és nagybetűre konvertálva visszaküldi a kliens UDP socketre.

```
  package org.ait;

  import java.net.DatagramPacket;
  import java.net.DatagramSocket;
  import java.net.InetAddress;

  public class UDPServer {
    public static void main(String args[]) throws Exception {

        DatagramSocket serverSocket = new DatagramSocket(8080);

        byte[] bytesReceived = new byte[1024];
        byte[] bytesSent = new byte[1024];


        DatagramPacket receivePacket = new DatagramPacket(bytesReceived,
bytesReceived.length);
```

```
        // itt várakozik ameddig adat jön a 8080-as porton
        serverSocket.receive(receivePacket);

        String szoveg = new String(receivePacket.getData());

        System.out.println("kaptam: " + szoveg);

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

        String nagybetűsSzöveg = szoveg.toUpperCase();
        bytesSent = nagybetűsSzöveg.getBytes();

        // visszaküldi
        DatagramPacket sendPacket = new DatagramPacket(bytesSent,
bytesSent.length, IPAddress, port);
        serverSocket.send(sendPacket);
        serverSocket.close();

    }
  }
```