Quiz 1 Prep

1. TCP/IP - Sockets

Varianta mai usoara de test

Se poate scrie entirely in main class, dar poate fi transpus usor si in clase care dau extend la Thread

```
public class Server {
  public static void main(String[] args) throws IOException{

    //Socket pentru server
    ServerSocket ss=null;
    //Socket pentru client
    Socket s=null;

    try{
      //Asta e pentru ce vrem sa transmitem
      String line="";

      //Setam port-ul/socket-ul pentru server socket
      ss = new ServerSocket(1900);
      System.out.println("Serverul asteapta conexiuni...");

      //Primeste socket-ul de la client si informatiile aferente
      s = ss.accept();

      //Basically pentru citit de la client
      BufferedReader in = new BufferedReader(
            new InputStreamReader(s.getInputStream()));

      //Face set up la un mod de a trimite la client informatii
      PrintWriter out = new PrintWriter(
            new BufferedWriter(new OutputStreamWriter(s.getOutputStream())),true);

      //Primeste adresa de la server client
      InetSocketAddress remoteadr = (InetSocketAddress)s.getRemoteSocketAddress();
      //Primeste remotehost si remoteport de la client (pentru comunicare)
      String remotehost = remoteadr.getHostName();
      int remoteport = remoteadr.getPort();

      System.out.println("Client nou conectat: "+remotehost+":"+remoteport);

      //Citeste de la tastatura :))
      line = in.readLine();
      System.out.println("Server a receptionat:"+line);

      //Scrie / trimite la server ceva (in cazul asta un string)
      out.println(result);
      //Avem nevoie de flush ca sa poata clientul receptiona
      out.flush();

      System.out.println("Aplicatie server gata.");

    }catch(Exception e){e.printStackTrace();}
     finally{
      ss.close();
      if(s!=null) s.close();
     }
```

```
  }
}
```

```
public class Client {

  public static void main(String[] args)throws Exception{
    Socket socket=null;
    try {
      //Nush de la care vine exception-ul, dar sunt mai multe, deeci :))
      //Set la server address, localhost in cazul asta
      InetAddress server_address = InetAddress.getByName("localhost");

      //Set la socket pentru conexiunea la server
      //ii dam server address si server socket-ul
      socket = new Socket(server_address, 1900);

      //La fel ca si la server, pentru read si write
      //prin in primeste de la server
      BufferedReader in =
          new BufferedReader(
              new InputStreamReader(
                  socket.getInputStream()));

      //prin out trimite la server
      PrintWriter out =
              new PrintWriter(
                new BufferedWriter(
                  new OutputStreamWriter(
                    socket.getOutputStream())),true);

      //pentru citirea de la tastatura
      BufferedReader reader = new BufferedReader(
                new InputStreamReader(System.in));

      //citeste din consola
      String output = reader.readLine();
      //trimite la server
      out.println(output);
      //primeste de la server
      String str = in.readLine();
      //printeaza ce a primit de la server
      System.out.println(str+"%");
      out.println("END");
    }
    catch(Exception ex) {
      ex.printStackTrace();
    }
    finally {
      socket.close();
    }
  }
}
```

TLDR rapid, poate ajuta:

Pentru TCP/IP :

Server:

- ServerSocket - pe ce functioneaza serverul
- Socket - stocheaza de unde primeste, deci socketul clientului

- BufferedReader - pentru a primi de la client
- PrintWriter - pentru a trimite la client

Client:

- Socket - al Clientului
- BufferedReader - pentru a primi de la server
- PrintWriter - pentru a trimite la server

2. UTP - Datagrams

Server/Receiver layout (or whatever)

```
public class Server extends Thread{
    protected DatagramSocket socket = null;
    //Not necessary but it's an example for an input
    protected BufferedReader in = null;

    //Default Constructor, not sure if needed, but can't hurt :))
    //Basically just gives a default name for the thread
    public Server() throws IOException {
        this("Server");
    }

    public Server(String name) throws IOException {
        super(name); //to set the name basic property of Thread
        socket = new DatagramSocket(4445); //sets the socket, number can be changed

        //Pana aici e default implementation, sa zicem ca folosim un txt file ca si input, continua de aici
        try {
            in = new BufferedReader(new FileReader("one-liners.txt"));
        } catch (FileNotFoundException e) {
            System.err.println("Could not open quote file. Serving time instead.");

        //Daca cere din consola, adaug si asta, ca stiu ca Java e retardaded cu sintaxa pentru
        //o simpla fucking citire (maybe just me :3 )
        in = new BufferedReader(
            new InputStreamReader(System.in));
        }
    }

    public void run() {

        while (true) { //sau alta conditie bazata pe situatie
            try {
                byte[] buf = new byte[256];

               // primeste request-ul de la Client
                DatagramPacket packet = new DatagramPacket(buf, buf.length);
                // din cauza ca e transmisie prin sockets, trebuie specificat cati bytes
                // trimitem
                // functioneaza ca si un notify pe thread, deci thread-ul de Server
                // va astepta pana cand primeste un packet nou

                // Primim raspunsul
                socket.receive(packet);
```

```
                // Facem un mumbo jumbo si trimitem ceva inapoi (daca vrem ofc)
                String dString = new Date().toString();
                buf = dString.getBytes();

                // Trimitem raspunsul inspre Client/Sender sau cum l-am numit

                //Prima data obtinem adresa si portul
                InetAddress address = packet.getAddress();
                int port = packet.getPort();

                // construim packet-ul nou
                packet = new DatagramPacket(buf, buf.length, address, port);
                // trimitem packet-ul
                socket.send(packet);
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        socket.close();
    }
}

//!!! Dam start la Server thread unde e nevoie, ori in clasa separata, ori in aceeasi
```

Client/Sender e mult mai simplu, din fericire

Codul din clasa de Client poate de asemenea sa fie scris in main, dar fac exemplul cu clasa separata din cauza ca e mai lung si mai general

```
class Client extends Thread{

  //Aceeasi explicatie ca si sus pentru constructor
  public Sender(String name) {
      super(name);
  }

  public void run() {
    try {
      // initializam socket-ul si setam adresa (in cazul asta localhost)
      DatagramSocket socket = new DatagramSocket();
      byte[] buf = new byte[256];
      InetAddress adress = InetAddress.getByName("localhost");

      // cream pachetul de trimis spre Server (tinem minte ce socket number am pus)
      DatagramPacket packet = new DatagramPacket(buf, buf.length, adress, 4445);
      // trimitem pachetul
      socket.send(packet);

      // primim pachetul nou de la server
      packet= new DatagramPacket(buf, buf.length);
      socket.receive(packet);

      // facem whatever the fuck we want cu datele primite
      String received = new String(packet.getData());
      outputArea.append("\n>"+received);

      socket.close();
    } catch (SocketException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
    } catch (UnknownHostException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
    } catch (IOException e) {
```

```
      // TODO Auto-generated catch block
      e.printStackTrace();
    }


  }
}
```

TLDR rapid, poate ajuta:

Server:

- DatagramSocket - pentru server
- Toate mesajele se transmit prin byte arrays
- Cand primim avem nevoie doar de buffer si length-ul buffer-ului
- Cand trimitem, avem nevoie additionally de adresa si de port pentru client (pe care le obtinem prin pachetul primit la input)

Client:

- DatagramSocket - pentru client
- Tot prin byte array trimitem si primim
- Tot aceeasi regula la primit si trimis