EXAM TP 2022

Table de multiplication UDP (7 points)

Un serveur UDP est lancé sur une machine, ce serveur écoute en UDP sur le port 11000.
Ce serveur fonctionne ainsi :
• il attend un paquet UDP contenant le texte « JOUER »
• quand il reçoit ce paquet avec JOUER, il répond avec un premier paquet UDP contenant un nombre compris entre 0 et 9 (par exemple 2), puis immédiatement après il envoie également un deuxième paquet UDP contenant un nombre compris entre 0 et 9 (par exemple 8)
• vous devez alors lui renvoyer le résultat de la multiplication des 2 nombres dans un paquet UDP , c'est à dire 16 dans notre exemple, avec un point-virgule à la fin
• dans notre exemple, vous devez donc retourner **16;**
• le serveur vous répond « GAGNE » ou « PERDU » si votre réponse est correcte ou non.
L'objectif de cet exercice est de réaliser un programme capable de jouer avec ce serveur.

```java
package Exam2022;

import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetSocketAddress;

public class ExolclientUDP {
    public static void main(String[] args) throws Exception
    {
        ExolclientUDP serveurUDP = new ExolclientUDP();
        serveurUDP.execute();
    }
    private void execute() throws IOException
    {
        System.out.println("Demarrage du client ...");

        //Creation de la socket
        DatagramSocket socket = new DatagramSocket();

        // Creation et envoi du message à l'adresse 127.0.0.1 et le port 11000
        InetSocketAddress adrDest = new InetSocketAddress("127.0.0.1", 11000);
        for(int i =1;i<=10;i++) {

            System.out.println("========================================");
            System.out.println("Début de la partie "+i);

            byte[] bufE = new String("JOUER").getBytes();//stockage lessage dans buffer
            DatagramPacket dpE = new DatagramPacket(bufE, bufE.length, adrDest); //envoie du message a l'adresse
            socket.send(dpE);
            System.out.println("Envoi d'un paquet UDP avec JOUER");

            // Le serveur se declare aupres de la couche transport
            // sur le port 11000
            //socket = new DatagramSocket(null);
            //socket.bind(new InetSocketAddress(11000));
            // Attente du premier message
            byte[] bufR = new byte[2048];
            DatagramPacket dpR = new DatagramPacket(bufR, bufR.length);
            socket.receive(dpR);//attente ler du message du serveur
            String message1 = new String(bufR, dpR.getOffset(), dpR.getLength());

            System.out.println("IP = "+dpR.getAddress().getHostAddress());
            System.out.println("Port = "+dpR.getPort());

            //System.out.println("Message recu = "+message1);
            String messplit[] = message1.split(";");
            Integer nb1 = Integer.parseInt(messplit[0]);

            //byte[] bufR_1 = new byte[2048];
            dpR = new DatagramPacket(bufR, bufR.length);
            socket.receive(dpR);//attente ler du message du serveur
            String message2 = new String(bufR, dpR.getOffset(), dpR.getLength());

            //System.out.println("Message recu = "+message2);

            String messplit2[] = message2.split(";");
```

```
58              Integer nb2 = Integer.parseInt(messplit2[0]);
59
60              String res= String.valueOf(nb1*nb2);
61
62              bufE = new String(res+";").getBytes();//stockage lessage dans buffer
63              System.out.println("Le serveur a répondu "+nb1+" et "+nb2);
64
65              dpE = new DatagramPacket(bufE, bufE.length, adrDest); //envoie du message a l'adresse
66              socket.send(dpE);
67              System.out.println("Envoi d'un paquet UDP avec "+res+";");
68
69              dpR = new DatagramPacket(bufR, bufR.length);
70              socket.receive(dpR);//attente ler du message du serveur
71              String message3 = new String(bufR, dpR.getOffset(), dpR.getLength());
72
73              System.out.println("Message recu :"+message3);
74              System.out.println("Fin de la partie "+i);
75          }
76          // Fermeture de la socket
77          socket.close();
78          System.out.println("");
79          System.out.println("Arret du client .");
80      }
81  }
```

Port scanning TCP (13 points)

Un serveur TCP est lancé sur une machine, ce serveur écoute en TCP sur **tous** les ports compris entre 30 000 et 32 000 (inclus).
Ce serveur fonctionne ainsi :

- le serveur écoute sur **tous** les ports compris entre 30 000 et 32 000 (inclus)
- quand un client se connecte sur un de ces ports, le serveur attend un message contenant une seule lettre : ?
- quand le serveur reçu cette lettre ? , **le serveur répond 5 secondes plus tard**
- la réponse du serveur peut être
  - VOUS AVEZ GAGNE!
  - VOUS AVEZ PERDU!
  - VOUS AVEZ FAIT UNE ERREUR.

Un seul numéro de port répond « VOUS AVEZ GAGNE! ».
Tous les autres numéros de port répondent « VOUS AVEZ PERDU! ».
Le serveur répond « VOUS AVEZ FAIT UNE ERREUR. » si la lettre envoyée par le client est différente de ?. L'objectif de cet exercice est de réaliser un programme capable de déterminer le port X correspondant à « VOUS AVEZ GAGNE! » et de l'afficher.

Port d'écoute : 31256

```java
package Exam2022;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.InetSocketAddress;
import java.net.Socket;

public class Exo2ClientTCP extends Thread{
    int port;
    public Exo2ClientTCP (int port){
        this.port=port;
    }
    public void run(){
        Exo2ClientTCP clientTCP = new Exo2ClientTCP(port);
        try {
            clientTCP.execute();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
    private void execute() throws IOException
    {
        //
        //System.out.println("Demarrage du client ...");

        //Creation de la socket
        Socket socket = new Socket();
        // Connexion au serveur
        InetSocketAddress adrDest = new InetSocketAddress("127.0.0.1", port);
        socket.connect(adrDest);
        // Envoi de la requete
        byte[] bufE = new String("?").getBytes();
        OutputStream os = socket.getOutputStream(); //permet d'envoyer dans
        os.write(bufE);
        //System.out.println("Message envoye");
        // Attente de la reponse
        byte[] bufR = new byte[2048];
        InputStream is = socket.getInputStream();
        String reponse ="";

        while (!reponse.contains("GAGNE") && (!reponse.contains("PERDU"))) {
            int lenBufR = is.read(bufR);
            if (lenBufR!=-1)
            {
                reponse = new String(bufR, 0 , lenBufR );
                //System.out.println("Reponse recue = "+reponse);
            }
        }
        if (reponse.contains("GAGNE")) {
            System.out.println("Début de la recherche ...");
            System.out.println("Le port d'écoute qui répond VOUS AVEZ GAGNE! est "+port);
            System.out.println("Fin du programme");
        }
        // Fermeture de la socket
        socket.close();

        //System.out.println("Arret du client .");
    }
    public static void main(String[] args) throws Exception {
        Exo2ClientTCP[] tab = new Exo2ClientTCP[2001];
        for (int i=0;i<=2000;i++) {//initialisation des thread
            Exo2ClientTCP newtcp = new Exo2ClientTCP(i+30000);
            tab [i] = newtcp;
        }
        for(int i= 0;i<2000;i++)//lancement des thread
        {
            tab[i].start();
        }
    }
}
```