ESISAR

NE449 - Programmation répartie

TDM numéro 2

SOLUTIONS

Table des matières

1 Solution 1

2 Solution 2

Il y a bien sûr plusieurs solutions, en voici une.

On fixe de façon arbitraire les ports de réception et des messages UDP :

- le terminal 1 écoute sur 3001
- le terminal 2 écoute sur 3002
- le terminal 3 écoute sur 3003
- le terminal 4 écoute sur 3004

Le programme 1 commence. Il affiche rouge pendant deux secondes, puis il envoie un paquet UDP au programme 2 (donc sur le port 3002).

Le programme 2 reçoit le paquet, se met en rouge , attend deux secondes, se met en vert, et envoie un paquet au programme 3 (donc sur le port 3003), et se remet à l'écoute.

Le programme 3 reçoit le paquet, se met en rouge , attend deux secondes, se met en vert, et envoi un paquet au programme 3 (donc sur le port 3003).

Votre programme doit donc avoir le comportement suivant :

- s'afficher en vert
- se mettre à l'écoute
- quand il reçoit un paquet UDP, se mettre en rouge
- attendre 2 secondes
- se mettre en vert
- envoyer un paquet UDP au suivant
- se re mettre à l'écoute

avec une variation pour le programme 1.

Le programme prend en entrée un seul paramètre, qui est le numéro du terminal, et adapte son comportement en conséquence.

Code :

```
import java.awt.Color;

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

import javax.swing.JFrame;
```

```
public class Main {


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

                int DEFAULT_PORT = 3000;
                int WINDOW_SIZE =  300;
                int windowNumber =Integer.valueOf(args[0]); // on recupere le numero de fenêtre des paramétres
                int currentPort = DEFAULT_PORT+ windowNumber;
                int nextPort = windowNumber==  4 ? DEFAULT_PORT+ 1 : currentPort +1 ;
            JFrame frame = new JFrame("Chenillard"+ currentPort);
        frame.setSize(300, 300);
        frame.setLocation(windowNumber*WINDOW_SIZE,windowNumber);
        frame.getContentPane().setBackground(Color.GREEN);
        frame.setVisible(true);


                while(true)
                {

                                DatagramSocket socket = new DatagramSocket(null);
                        socket = new DatagramSocket(null);
                        socket.bind(new InetSocketAddress(currentPort));
                    // Attente du premier message
                    byte[] bufR = new byte[2048];
                    DatagramPacket dpR = new DatagramPacket(bufR, bufR.length);
                    socket.receive(dpR);
                    String message = new String(bufR, dpR.getOffset(), dpR.getLength());
                    System.out.println("Message recu = "+message);

                    socket.close();
                    frame.getContentPane().setBackground(Color.RED);
                    frame.setVisible(true);
```

```
                Thread.sleep(1000);

                //
                frame.getContentPane().setBackground(Color.GREEN);
                frame.setVisible(true);



                socket = new DatagramSocket();

                //Envoi d'un message au suivant
                InetSocketAddress adrDest = new InetSocketAddress("127.0.0.1", nextPort);
                byte[] bufE = new String("hello").getBytes();
                DatagramPacket dpE = new DatagramPacket(bufE, bufE.length, adrDest);
                socket.send(dpE);
                System.out.println("Message envoyé."+ nextPort);

                socket.close();
            }

        }

}
```

compilation :

javac Main.java

Lancement des 4 fenêtres :

java Main 1 &

java Main 2 &

java Main 3 &

java Main 4 &

puis on envoie un message à la première fenetre :

nc -u 127.0.0.1 3001 && test

Variante avec des threads :

Main.java :

```
import java.awt.Color;

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

import javax.swing.JFrame;


public class Main {

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


		Chenille second = new Chenille(2, 3);

		Chenille third = new Chenille(3, 4);
		Chenille fourth = new Chenille(4, 1);
		Chenille one = new Chenille(1, 2);

		second.start();


		third.start();

		fourth.start();
```

```
                one.start();
                //test.start();
                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 2000
                InetSocketAddress adrDest = new InetSocketAddress("127.0.0.1", 2000);
                byte[] bufE = new String("hello").getBytes();
                DatagramPacket dpE = new DatagramPacket(bufE, bufE.length, adrDest);
                socket.send(dpE);
                System.out.println("Message envoyé.");

                // Fermeture de la socket
                socket.close();
                System.out.println("Arret du client .");

        }

}
```

Chenille.java :

```
import java.awt.Color;

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

```
import javax.swing.JFrame;



public class Chenille extends Thread {
        int WINDOW_SIZE = 300;
        int DEFAULT_PORT = 3000;

   int currentPort ;
   int nextPort ;
   JFrame frame;
   public Chenille( int currentPort, int nextPort) {
      this.currentPort = currentPort + DEFAULT_PORT;
      this.nextPort = nextPort+ DEFAULT_PORT;


      JFrame frame = new JFrame("Chenillard"+ this.currentPort);
      frame.setSize(WINDOW_SIZE, WINDOW_SIZE);
      frame.setLocation(currentPort*WINDOW_SIZE,currentPort);
      frame.getContentPane().setBackground(Color.GREEN);
      frame.setVisible(true);

      this.frame = frame;


   }


   public void run() {
        DatagramSocket socket =null;


      while(true)
      {
          try {
                socket = new DatagramSocket(null);
                socket.bind(new InetSocketAddress(this.currentPort));

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


        System.out.println("waiting."+ this.currentPort);
```

```
// Attente du premier message
byte[] bufR = new byte[2048];
DatagramPacket dpR = new DatagramPacket(bufR, bufR.length);
try {
                socket.receive(dpR);
        } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }


String message = new String(bufR, dpR.getOffset(), dpR.getLength());
System.out.println("Message recu = "+message);

socket.close();
this.frame.getContentPane().setBackground(Color.RED);
this.frame.setVisible(true);

//On attend une seconde
try {
                Thread.sleep(1000);
        } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }

//on repasse en vert

this.frame.getContentPane().setBackground(Color.GREEN);
this.frame.setVisible(true);

//Preparation pour envoyer le message

    DatagramSocket socket2 =null;

try {
                socket2 = new DatagramSocket();
        } catch (SocketException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }

//Envoi d'un message au suivant
InetSocketAddress adrDest = new InetSocketAddress("127.0.0.1", this.nextPort);
byte[] bufE = new String("hello").getBytes();
DatagramPacket dpE = new DatagramPacket(bufE, bufE.length, adrDest);
try {
```

```
                        socket2.send(dpE);
                    } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    }
            System.out.println("Message envoyé."+ this.nextPort);

           socket2.close();
        }

    }

}
```

3 Solution 3

Code client :

```
import java.awt.Color;

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

import javax.swing.JFrame;


public class Client {


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

                int DEFAULT_PORT = 3000;
                int WINDOW_SIZE =  300;
```

```
            int windowNumber =Integer.valueOf(args[0]); // on recupere le numero de
fenêtre des paramétres

            boolean isLast = args.length == 2;



            int currentPort = DEFAULT_PORT+ windowNumber;

        JFrame frame = new JFrame("Chenillard"+ currentPort);
      frame.setSize(300, 300);
      frame.setLocation(windowNumber*WINDOW_SIZE,windowNumber);
      frame.getContentPane().setBackground(Color.GREEN);
      frame.setVisible(true);



    DatagramSocket socketReceive = new DatagramSocket(null);
            socketReceive = new DatagramSocket(null);
            socketReceive.bind(new InetSocketAddress(currentPort));
             // Attente du premier message
      byte[] bufR = new byte[2048];
      DatagramPacket dpR = new DatagramPacket(bufR, bufR.length);

      //send message au serveur
      DatagramSocket socket = new DatagramSocket();
      InetSocketAddress adrDest = new InetSocketAddress("127.0.0.1", DEFAULT_PORT);
      String messageSend = String.valueOf( currentPort);
      if(isLast)
      {
        messageSend = String.valueOf( currentPort)+ "_dernier";
      }
      byte[] bufE = new String(messageSend).getBytes();
      DatagramPacket dpE = new DatagramPacket(bufE, bufE.length, adrDest);
      socket.send(dpE);
      System.out.println("Message envoyé au serveur ."+ currentPort + " "+
messageSend);
```

```
        socket.close();


            socketReceive.receive(dpR);

               while(true)
               {



                  String message = new String(bufR, dpR.getOffset(), dpR.getLength());
                  System.out.println("Message recu du servuer = "+message);

                  if(message.equals("green"))
                  {
                     frame.getContentPane().setBackground(Color.GREEN);
                          frame.setVisible(true);
                  }else {
                     frame.getContentPane().setBackground(Color.RED);
                          frame.setVisible(true);
                  }
                  socketReceive.receive(dpR);









               }

         }

}
```

code serveur :

```
import java.awt.Color;
import java.awt.List;
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetSocketAddress;
import java.net.SocketException;
import java.util.ArrayList;
import java.io.IOException;

import javax.swing.JFrame;


public class Serveur {


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


		int currentPort = 3000;
		ArrayList<Integer> clientPorts = new ArrayList<>();

	  DatagramSocket socket = new DatagramSocket(null);
	socket = new DatagramSocket(null);
	socket.bind(new InetSocketAddress(currentPort));


	boolean isWaitingForClients = true;
	while(isWaitingForClients)
	{

	   // Attente du premier message
	   byte[] bufR = new byte[2048];
```

```
    DatagramPacket dpR = new DatagramPacket(bufR, bufR.length);
    socket.receive(dpR);
    String message = new String(bufR, dpR.getOffset(), dpR.getLength());
    System.out.println("Message recu = "+message);
   System.out.println("Port = "+dpR.getPort());


    String[] splitMessage = message.split("_");
    if(splitMessage.length == 2)
    {
    System.out.println("Dernier reçu = "+dpR.getPort());


            isWaitingForClients = false;

    }

   int clientPort= Integer.valueOf(splitMessage[0]);
    clientPorts.add(clientPort);




}

while(true)
{
  for(int i = 0; i < clientPorts.size(); i++)
{
  sendMessage(clientPorts.get(i), "red");

  Thread.sleep(1000);
  sendMessage(clientPorts.get(i), "green");

}
```

```
            }

            // socket.close();

        }

         public static void sendMessage(int portDest, String message) throws
SocketException,IOException
    {
        DatagramSocket socket = new DatagramSocket();

       //Envoi d'un message au suivant
       InetSocketAddress adrDest = new InetSocketAddress("127.0.0.1", portDest);
       byte[] bufE = new String(message).getBytes();
       DatagramPacket dpE = new DatagramPacket(bufE, bufE.length, adrDest);
       socket.send(dpE);
       System.out.println("Message envoyé."+ portDest);

       socket.close();

    }

}
```

Javac Client.java

javac Serveur.java

On lance le serveur d'abord :

java Serveur

```
 java Client 4 &
 java Client 5 1 => pour indiquer que c'est le dernier
```

4 Solution 4