

## UF1.NF1.Sòcols

1.- Crea una classe SocketUDP que contingui dos ports: un per enviar i un per escoltar. Crea els constructors i accessors pertinents.

```
public class SocketUDP {  
    4 usages  
    private int portEnviar;  
    4 usages  
    private int portEscoltar;  
  
    public SocketUDP(){  
        this.portEnviar = 6666;  
        this.portEscoltar = 5555;  
    }  
  
    public int getPortEnviar() {  
        return portEnviar;  
    }  
  
    public void setPortEnviar(int portEnviar) {  
        this.portEnviar = portEnviar;  
    }  
  
    public int getPortEscoltar() {  
        return portEscoltar;  
    }  
  
    public void setPortEscoltar(int portEscoltar) {  
        this.portEscoltar = portEscoltar;  
    }  
}
```

2.- Crea un mètode enviarMissatge a la classe SocketUDP que rebi com a paràmetres dos Strings: els missatge i la IP del destí, i l'envii al sòcol de la IP.

```
public void enviarMissatge(String ip, String missatge){  
    try {  
        DatagramSocket socket = new DatagramSocket();  
  
        byte[] mensaje = missatge.getBytes();  
  
        InetAddress adrecaDesti = InetAddress.getByName(ip);  
        DatagramPacket paquet = new DatagramPacket(mensaje, mensaje.length, adrecaDesti, portEnviar);  
  
        socket.send(paquet);  
        socket.close();  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
}
```

3.- Crea un mètode `rebreMissatge` a la classe `SocketUDP` que rebí el paquet provinent del port que escolta i en mostri la IP i el port.

```
public void rebreMissatge() {
    try {
        DatagramSocket socket = new DatagramSocket(portEscoltar);

        byte[] buffer = new byte[1024];
        DatagramPacket paquet = new DatagramPacket(buffer, buffer.length);

        socket.receive(paquet);

        String missatge = new String(paquet.getData(), 0, paquet.getLength());
        InetAddress adrecaEnviador = paquet.getAddress();
        int portEnviador = paquet.getPort();

        System.out.println("Missatge rebut: " + missatge);
        System.out.println("IP de l'enviador: " + adrecaEnviador.getHostAddress());
        System.out.println("Port de l'enviador: " + portEnviador);

        socket.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

4.- Instancia dos sòcols i simula l'enviament del paquet amb els mètodes creats anteriorment.

```
public class Main {
    public static void main(String[] args) {
        SocketUDP socket1 = new SocketUDP(1234, 5432);
        SocketUDP socket2 = new SocketUDP(5432, 1234);

        String missatge1 = "Hola, sòcol 2!";
        String adrecaDesti1 = "localhost";

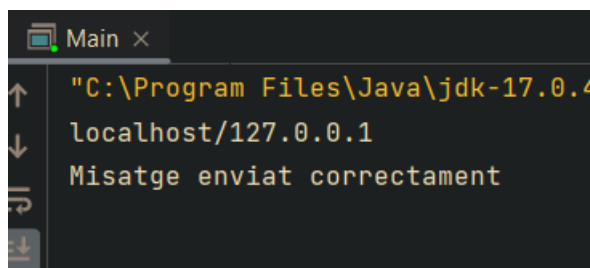
        socket1.enviarMissatge(adrecaDesti1, missatge1);
    }
}
```

5.- Modifica l'exercici anterior perquè mostri també el missatge que s'envia.

```
public class Main {  
    public static void main(String[] args) {  
        SocketUDP socket1 = new SocketUDP(1234, 5432);  
        SocketUDP socket2 = new SocketUDP(5432, 1234);  
  
        String missatge1 = "Hola, sòcol 2!";  
        String adrecaDesti1 = "localhost";  
  
        socket1.enviarMissatge(adrecaDesti1, missatge1);  
        socket2.rebreMissatge();  
    }  
}
```

6.- Simula que dos sòcols escolten a la mateixa adreça i mateix port. Què passa?

```
public class Main {  
    public static void main(String[] args) {  
        SocketUDP socket1 = new SocketUDP(1234, 1234);  
        SocketUDP socket2 = new SocketUDP(1234, 1234);  
  
        String missatge1 = "Hola, sòcol 2!";  
        String adrecaDesti1 = "localhost";  
  
        socket1.enviarMissatge(adrecaDesti1, missatge1);  
        socket2.rebreMissatge();  
    }  
}
```



```
Main x  
"C:\Program Files\Java\jdk-17.0.4  
localhost/127.0.0.1  
Missatge enviat correctament
```

7.- Modifica l'exercici 5 de tal manera que els dos sòcols rebin i envïin de forma permanent fins que l'usuari escrigui el caràcter '/'.

```
public class Main {  
    public static void main(String[] args) {  
        SocketUDP socket1 = new SocketUDP(1234, 4321);  
        SocketUDP socket2 = new SocketUDP(4321, 1234);  
  
        String missatge1 = "Hola, sòcol 2!";  
        String adrecaDesti1 = "localhost";  
  
        socket1.enviarMissatge(adrecaDesti1, missatge1);  
        socket2.rebreMissatge();  
  
        Scanner scanner = new Scanner(System.in);  
        while (true) {  
            String missatge = scanner.nextLine();  
            if (missatge.equals("/")) {  
                break;  
            }  
        }  
    }  
}
```

8.- Modifica l'exercici 7 perquè el sòcol que escolta, només ho faci durant 3 segons. Si no ha rebut res en 3 segons, s'ha de tancar el sòcol i finalitzar el mètode.

```
public void rebreMissatge() {  
    try {  
        DatagramSocket socket = new DatagramSocket(portEscoltar);  
        socket.setSoTimeout(3000);  
  
        byte[] buffer = new byte[1024];  
        DatagramPacket paquet = new DatagramPacket(buffer, buffer.length);  
  
        socket.receive(paquet);  
  
        String missatge = new String(paquet.getData(), 0, paquet.getLength());  
        InetAddress adrecaEnviador = paquet.getAddress();  
        int portEnviador = paquet.getPort();  
  
        System.out.println("Missatge rebut: " + missatge);  
        System.out.println("IP de l'enviador: " + adrecaEnviador.getHostAddress());  
        System.out.println("Port de l'enviador: " + portEnviador);  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
}
```



9.- Simula l'enviament d'un missatge de servidor a client però utilitzant sòcols multicast. Si poses diferents ports què passa?

```
public void enviarMissatgeMultiSocket(String ip, String missatge){
    try {
        MulticastSocket socket = new MulticastSocket(portEnviar);

        byte[] mensaje = missatge.getBytes();

        InetAddress adrecaDesti = InetAddress.getByName(ip);
        DatagramPacket paquet = new DatagramPacket(mensaje, mensaje.length, adrecaDesti, portEnviar);

        socket.send(paquet);

        System.out.println("Missatge enviat correctament");
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

```
public void rebreMissatgeMultiSocket() {
    try {
        MulticastSocket socket = new MulticastSocket(portEscoltar);

        NetworkInterface net = NetworkInterface.getByNames("localhost");

        InetSocketAddress group = new InetSocketAddress(InetAddress.getByName("127.0.0.1"), portEscoltar);
        socket.joinGroup(group, net);

        byte[] buffer = new byte[1024];
        DatagramPacket paquet = new DatagramPacket(buffer, buffer.length);

        socket.receive(paquet);

        String missatge = new String(paquet.getData(), 0, paquet.getLength());
        InetAddress adrecaEnviador = paquet.getAddress();
        int portEnviador = paquet.getPort();

        System.out.println("Missatge rebut: " + missatge);
        System.out.println("IP de l'enviador: " + adrecaEnviador.getHostAddress());
        System.out.println("Port de l'enviador: " + portEnviador);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

10.- Modifica l'exercici anterior perquè dos sòcols multicast rebin simultàniament el missatge del sòcol servidor.

11.- Crea una classe SocketTCPClient i SocketTCPServidor i simula l'enviament d'un paquet de client a servidor.

```
public class SocketTCPClient {
    public static void main(String[] args) {
        final String servidorIP = "localhost";
        final int port = 12345;

        try {
            Socket clientSocket = new Socket(servidorIP, port);
            System.out.println("Connectat al servidor " + servidorIP + " al port " + port);

            PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);
            String missatgeAEnviar = "Aquest és un missatge de prova del client";
            out.println(missatgeAEnviar);

            out.close();
            clientSocket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

```
public class SocketTCPServidor {
    public static void main(String[] args) {
        final int port = 12345;

        try {
            ServerSocket serverSocket = new ServerSocket(port);

            System.out.println("Servidor en línia. Esperant connexions...");

            Socket clientSocket = serverSocket.accept();
            System.out.println("Client connectat des de: " + clientSocket.getInetAddress());

            BufferedReader in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
            String missatgeRebut = in.readLine();
            System.out.println("Missatge rebut del client: " + missatgeRebut);

            in.close();
            clientSocket.close();
            serverSocket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

```
SocketTCPServidor x SocketTCPClient x
"C:\Program Files\Java\jdk-17.0.4.1\bin\java.exe" "-javaagent:C:\Progr
Servidor en línia. Esperant connexions...
Client connectat des de: /127.0.0.1
Missatge rebut del client: Aquest és un missatge de prova del client

Process finished with exit code 0
```

```
SocketTCPServidor x SocketTCPClient x
"C:\Program Files\Java\jdk-17.0.4.1\bin\java.exe
Connectat al servidor localhost al port 12345
|
Process finished with exit code 0
```

## 12.- Modifica l'exercici anterior per enviar un paquet del servidor al client.

```
public class SocketTCPClient {
    public static void main(String[] args) {
        final String servidorIP = "localhost";
        final int port = 12345;

        try {
            Socket clientSocket = new Socket(servidorIP, port);
            System.out.println("Connectat al servidor " + servidorIP + " al port " + port);

            PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);
            String missatgeAEnviar = "Aquest és un missatge de prova del client";
            out.println(missatgeAEnviar);

            BufferedReader in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
            String respostaServidor = in.readLine();
            System.out.println("Resposta del servidor: " + respostaServidor);

            in.close();
            out.close();
            clientSocket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

```
public class SocketTCPServidor {
    public static void main(String[] args) {
        final int port = 12345;

        try {
            ServerSocket serverSocket = new ServerSocket(port);

            System.out.println("Servidor en línia. Esperant connexions...");

            Socket clientSocket = serverSocket.accept();
            System.out.println("Client connectat des de: " + clientSocket.getInetAddress());

            BufferedReader in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
            String missatgeRebut = in.readLine();
            System.out.println("Missatge rebut del client: " + missatgeRebut);

            PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);
            String respostaServidor = "Aquesta és una resposta del servidor";
            out.println(respostaServidor);

            in.close();
            out.close();
            clientSocket.close();
            serverSocket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```



```
SocketTCPServidor x SocketTCPClient x
" C:\Program Files\Java\jdk-17.0.4.1\bin\java.exe" "-javaagent:C:\Progra
Servidor en línia. Esperant connexions...
Client connectat des de: /127.0.0.1
Missatge rebut del client: Aquest és un missatge de prova del client

Process finished with exit code 0
```

```
SocketTCPServidor x SocketTCPClient x
" C:\Program Files\Java\jdk-17.0.4.1\bin\java.exe" "-javaagent
Connectat al servidor localhost al port 12345
Resposta del servidor: Aquesta és una resposta del servidor

Process finished with exit code 0
```

**13.- Fes el mètodes d'enviament de paquets utilitzant sòcols TCP de manera permanent (com si fos un xat). Fes la prova amb un company posant la seva IP i port i a la inversa. Un haurà de fer de servidor i l'altre de client.**