Escribe una pareja de programas cliente/servidor usando sockets stream en el que el servidor esté constantemente a la espera de recibir conexiones y cuando reciba una muestre por pantalla la dirección IP del cliente que se ha conectado y lleve una cuenta del número de clientes que se han ido conectando. El servidor enviará un mensaje al cliente indicándole en número de cliente que es y el cliente mostrará por pantalla dicho mensaje

```
public class Servidor {
    static final int PUERTO = 5000;
    private int clientes = 0;

    public Servidor(){
        try{
            // Se crea el socket del servidor
            ServerSocket server = new ServerSocket(PUERTO);
            System.out.println("Servidor escuchando en puerto " + PUERTO);
            while (true) {
                // Se espera una conexión de un cliente
                Socket cliente = server.accept();
                // se incrementa el número de clientes
                clientes++;
                // cliente.getInetAddress() devuelve la dirección IP del cliente
                System.out.println("Cliente " + clientes + " IP: " + cliente.getInetAddress());
                // Se envía un mensaje al cliente
                DataOutputStream salida = new DataOutputStream(cliente.getOutputStream());
                salida.writeUTF("Hola cliente n°" + clientes);
                // Se cierra la conexión con el cliente
                cliente.close();
            }
        } catch(RuntimeException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
}
    public static void main(String[] args) {
        new Servidor();
    }
}
```

CON HILOS

```
public class Servidor extends Thread {
    static final int PUERTO = 5000;
    private static int clientes = 0;
    private Socket cliente;

    public Servidor(Socket cliente) {
        this.cliente = cliente;
    }

    public void run() {
        try {
                // se incrementa el número de clientes
                clientes++;
                // cliente.getInetAddress() devuelve la dirección IP del cliente
                System.out.println("Cliente " + clientes + " IP: " + cliente.getInetAddress());
                // Se envía un mensaje al cliente
                DataOutputStream salida = new DataOutputStream(cliente.getOutputStream());
                salida.writeUTF("Hola cliente n°" + clientes);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public static void main(String[] args) {
        try{
            // Se crea el socket del servidor
            ServerSocket server = new ServerSocket(PUERTO);
            System.out.println("Servidor escuchando en puerto " + PUERTO);
            while (true) {
                // Se espera una conexión de un cliente
                Socket cliente = server.accept();
```

```
                new Servidor(cliente).start();
            }
        } catch(RuntimeException e) {
                e.printStackTrace();
        } catch(IOException e) {
                e.printStackTrace();
        }
    }
}
```

```
public class Cliente {
    static final String HOST = "localhost";
    static final int PUERTO = 5000;

    public Cliente() {
        try {
            // Se crea el socket del cliente
            Socket cliente = new Socket(HOST, PUERTO);
            // Se recibe un mensaje del servidor
            DataInputStream entrada = new DataInputStream(cliente.getInputStream());
            // Se imprime el mensaje recibido
            System.out.println(entrada.readUTF());
            // Se cierra la conexión con el servidor
            cliente.close();
        } catch (RuntimeException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    public static void main(String[] args) {
        new Cliente();
    }
}
```

```
El objetivo del ejercicio es crear una aplicación cliente/servidor usando sockets
datagram que se comunique por el puerto 2000 y realice lo siguiente:
El servidor debe generar un número secreto de forma aleatoria entre el 0 al 100.
El objetivo del cliente es solicitarle al usuario un número y enviarlo al servidor
hasta que adivine el número secreto. Para ello, el servidor para cada número que
le envía el cliente le indicará si es menor, mayor o es el número secreto del
servidor.
```

```
public class ServidorUdp {
    static final int PUERTO = 2000;

    public static void main(String [] args) {
        int n = (int)(Math.random()*100)+1;
        String num = Integer.toString(n);
        System.out.println("Numero generado: " + num);
        try{
            // Se crea el socket del servidor
            DatagramSocket server = new DatagramSocket(PUERTO);
            InetAddress ip = InetAddress.getByName("localhost");
            // Crea el espacio para los mensajes
            byte[] recibe = new byte[1000];
            DatagramPacket mensaje = new DatagramPacket(recibe, recibe.length);
            System.out.println("Esperando mensajes..");
            boolean correcto = false;
            String res = "";
            while(!correcto){
                // Recibe el mensaje
                server.receive(mensaje);
                // convierte el mensaje de byte a String
                String datos = new String(mensaje.getData(), 0, mensaje.getLength());
                // convierte el String a int
                int numero = Integer.parseInt(datos);
                if(numero == n){
                    res = "Numero Correcto";
```

```
                    correcto = true;
                }else if(numero > n){
                    res = "numero Incorrecto, el numero secreto es menor";
                }else{
                    res = "numero Incorrecto, el numero secreto es mayor";
                }
                // envia la respuesta convertida a byte
                byte[] enviar = res.getBytes();
                // crea el paquete con la respuesta
                DatagramPacket respuesta = new DatagramPacket(enviar, enviar.length, ip,
mensaje.getPort());
                // envia la respuesta
                server.send(respuesta);
            }
            // cuando se adivina el numero secreto se cierra el socket y se termina el programa
            server.close();
        }catch (IOException e) {
            e.printStackTrace();
        }catch (RuntimeException e) {
            e.printStackTrace();
```

CON HILOS

```
public class ServidorUdp extends Thread {
    static final int PUERTO = 2000;
    private InetAddress ip;
    private int n;
    private DatagramPacket mensaje;
    private DatagramSocket server;

    public ServidorUdp(DatagramSocket server, DatagramPacket mensaje, InetAddress ip, int n){
        this.server = server;
        this.mensaje = mensaje;
        this.ip = ip;
        this.n = n;
    }
    public void run(){
        boolean correcto = false;
        String res = "";
        try{
            while(!correcto){
                // Recibe el mensaje
                server.receive(mensaje);
                // convierte el mensaje de byte a String
                String datos = new String(mensaje.getData(), 0, mensaje.getLength());
                // convierte el String a int
                int numero = Integer.parseInt(datos);
                if(numero == n){
                    res = "Numero Correcto";
                    correcto = true;
                }else if(numero > n){
                    res = "numero Incorrecto, el numero secreto es menor";
                }else{
                    res = "numero Incorrecto, el numero secreto es mayor";
                }
                // envia la respuesta convertida a byte
                byte[] enviar = res.getBytes();
                // crea el paquete con la respuesta
                DatagramPacket respuesta = new DatagramPacket(enviar, enviar.length, ip,
mensaje.getPort());
                // envia la respuesta
                server.send(respuesta);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public static void main(String [] args) {
        int n = (int)(Math.random()*100)+1;
        String num = Integer.toString(n);
        System.out.println("Numero generado: " + num);
        try{
            // Se crea el socket del servidor
```

```
            DatagramSocket server = new DatagramSocket(PUERTO);
            InetAddress ip = InetAddress.getByName("localhost");
            // Crea el espacio para los mensajes
            byte[] recibe = new byte[1000];
            DatagramPacket mensaje = new DatagramPacket(recibe, recibe.length);
            System.out.println("Esperando mensajes..");
            new ServidorUdp(server, mensaje, ip, n).start();
            // cuando se adivina el numero secreto se cierra el socket y se termina el programa
            // server.close();
        }catch (IOException e) {
            e.printStackTrace();
        }catch (RuntimeException e) {
            e.printStackTrace();
        }
    }
}
```

```
public class ClienteUdp {
    static final int PUERTO = 2000;

    public static void main(String [] args) {
        boolean fin = false;
        Scanner t = new Scanner(System.in);
        int intento = 1;
        DatagramSocket sSocket = null;
        do{
            System.out.println("Adivine el numero secreto (Intento: "+intento+")");
            try {
                String num = t.nextLine();
                // Crear el socket
                sSocket = new DatagramSocket();
                // construye la dirección del receptor pasando por argumento la máquina destino
                InetAddress maquina = InetAddress.getByName("localhost");
                // construye el mensaje
                byte[] cadena = num.getBytes();
                DatagramPacket mensaje = new DatagramPacket(cadena, num.length(), maquina,
PUERTO);
                // envía el mensaje
                sSocket.send(mensaje);
                // Crea el espacio para los mensajes
                byte[] recibe = new byte[1000];
                //  crea el paquete con la respuesta
                DatagramPacket respuesta = new DatagramPacket(recibe, recibe.length);
                // Recibe el mensaje
                sSocket.receive(respuesta);
                // convierte el mensaje de byte a String
                String datos = new String(respuesta.getData(), 0, respuesta.getLength());
                if(datos.equals("Numero Correcto")) {
                    System.out.println(datos);
                    System.out.println("Felicidades, ha adivinado el numero secreto");
                    fin = true;
                }else{
                    System.out.println(datos);
                    intento++;
                }
            }  catch(IOException e) {
                System.err.println("E/S: " + e.getMessage());

            }catch (RuntimeException e) {
                e.printStackTrace();
            }
        }while(!fin);
            // cierra el socket
            sSocket.close();
    }
}
```

```
El objetivo del ejercicio es crear una aplicación cliente/servidor usando sockets
stream que permita el envío de ficheros al cliente. Para ello, el cliente se
conectará al servidor por el puerto 1500 y le solicitará el nombre de un fichero del
```

servidor. Si el fichero existe, el servidor, le enviará el fichero al cliente y éste lo mostrará por pantalla. Si el fichero no existe, el servidor le enviará al cliente un mensaje de error. Una vez que el cliente ha mostrado el fichero se finalizará la conexión.

```java
public class Servidor {
    static final int PUERTO = 1500;

    public Servidor(){
        try{
            // Se crea el socket del servidor
            ServerSocket server = new ServerSocket(PUERTO);
            System.out.println("Servidor escuchando en puerto " + PUERTO);
            while (true) {
                // Se espera una conexión de un cliente
                Socket cliente = server.accept();
                // cliente.getInetAddress() devuelve la dirección IP del cliente
                System.out.println(" IP: " + cliente.getInetAddress());
                // Se envía un mensaje al cliente
                DataOutputStream salida = new DataOutputStream(cliente.getOutputStream());
                // salida.writeUTF escribe un mensaje en el flujo de salida
                salida.writeUTF("Conectado correctamente");
                boolean ok = false;
                while (!ok) {
                    // Se recibe un mensaje del cliente
                    DataInputStream entrada = new DataInputStream(cliente.getInputStream());
                    // guarda el nombre del archivo que el cliente quiere ver
                    String nombreArchivo = entrada.readUTF();
                    // imprime el nombre del archivo
                    System.out.println(nombreArchivo);
                    // convertir el nombre del archivo en un objeto Path
                    // para poder convertirlo en un archivo File y poder manipularlo
                    Path path = Path.of("src", nombreArchivo);
                    File file = new File(path.toString());
                    // se comprueba si el archivo existe
                    if (file.exists()) {
                        // con Files.readAllLines se lee el archivo y se guarda en una lista
                        List<String> mensaje = Files.readAllLines(path);
                        // se crea un string con el contenido del archivo
                        String msg = "";
                        for (String linea : mensaje) {
                            msg += linea + "\n";
                        }
                        // se envía el contenido del archivo al cliente
                        salida.writeUTF(msg);
                        // se cambia el valor de la variable para salir del bucle
                        ok = true;
                    } else {
                        // si el archivo no existe se envía un mensaje de error al cliente
                        salida.writeUTF("El archivo no existe");
                    }
                }
                // Se cierra la conexión con el cliente
                cliente.close();
            }
        } catch (EOFException e) {
            e.printStackTrace();
        } catch(RuntimeException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    public static void main(String[] args) {
        new Servidor();
    }
}
```

CON HILOS

```java
public class Servidor extends Thread {
    static final int PUERTO = 1500;
```

```
    private Socket cliente;
    private DataOutputStream salida;

    public Servidor(Socket cliente, DataOutputStream salida) {
        this.cliente = cliente;
        this.salida = salida;
    }
    public void run() {
        try {
            boolean ok = false;
            while (!ok) {
                // Se recibe un mensaje del cliente
                DataInputStream entrada = new DataInputStream(cliente.getInputStream());
                // guarda el nombre del archivo que el cliente quiere ver
                String nombreArchivo = entrada.readUTF();
                // imprime el nombre del archivo
                System.out.println(nombreArchivo);
                // convertir el nombre del archivo en un objeto Path
                // para poder convertirlo en un archivo File y poder manipularlo
                Path path = Path.of("src", nombreArchivo);
                File file = new File(path.toString());
                // se comprueba si el archivo existe
                if (file.exists()) {
                    // con Files.readAllLines se lee el archivo y se guarda en una lista
                    List<String> mensaje = Files.readAllLines(path);
                    // se crea un string con el contenido del archivo
                    String msg = "";
                    for (String linea : mensaje) {
                        msg += linea + "\n";
                    }
                    // se envía el contenido del archivo al cliente
                    salida.writeUTF(msg);
                    // se cambia el valor de la variable para salir del bucle
                    ok = true;
                } else {
                    // si el archivo no existe se envía un mensaje de error al cliente
                    salida.writeUTF("El archivo no existe");
                }
            }
            // Se cierra la conexión con el cliente
            cliente.close();
        } catch (EOFException e) {
            e.printStackTrace();
        } catch (RuntimeException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    public static void main(String[] args) {
        try {
            // Se crea el socket del servidor
            ServerSocket server = new ServerSocket(PUERTO);
            System.out.println("Servidor escuchando en puerto " + PUERTO);
            while (true) {
                // Se espera una conexión de un cliente
                Socket cliente = server.accept();
                // cliente.getInetAddress() devuelve la dirección IP del cliente
                System.out.println(" IP: " + cliente.getInetAddress());
                // Se envía un mensaje al cliente
                DataOutputStream salida = new DataOutputStream(cliente.getOutputStream());
                // salida.writeUTF escribe un mensaje en el flujo de salida
                salida.writeUTF("Conectado correctamente");
                new Servidor(cliente, salida).start();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

```java
public class Cliente {
    Scanner t = new Scanner(System.in);
    static final String HOST = "localhost";
    static final int PUERTO = 1500;

    public Cliente() {
        try {
            // Se crea el socket del cliente
            Socket cliente = new Socket(HOST, PUERTO);
            // Se recibe un mensaje del servidor
            DataInputStream entrada = new DataInputStream(cliente.getInputStream());
            // Se imprime el mensaje recibido
            System.out.println(entrada.readUTF());
            // Se envia un nombre de archivo al servidor para ver si existe
            String mensaje;
            do {
                System.out.println("Introduce el nombre del archivo que deseas ver: ");
                String nombreArchivo = t.nextLine();
                nombreArchivo += ".txt";
                DataOutputStream salida = new DataOutputStream(cliente.getOutputStream());
                salida.writeUTF(nombreArchivo);
                // Se recibe un mensaje del servidor
                System.out.println("Contenido del archivo: ");
                mensaje = entrada.readUTF();
                System.out.print(mensaje);
            }while (mensaje.equalsIgnoreCase("El archivo no existe"));
            // Se cierra la conexión con el servidor
            cliente.close();
        } catch (RuntimeException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    public static void main(String[] args) {
        new Cliente();
    }
}
```

```
Escribe un programa que conteste a preguntas.
El programa creará sockets datagram y esperará conexiones. Cuando llegue una
conexión, leerá los mensajes recibidos.
• Si la frase es “como te llamas”, responderá con una cadena de texto que
indique vuestro nombre.
• Si la frase es “cuál es tu edad”, responderá con vuestra edad.
• Si la frase es cualquier otra cosa, responderá “No he entendido la pregunta”
Escribe un programa cliente que haga preguntas al servidor para verificar su
funcionamiento.
Finalizará cuando el cliente envíe la palabra “fin
```

```java
public class Servidor {
    static final int PUERTO = 2000;

    public static void main(String [] args){
        try{
            // Se crea el socket del servidor
            DatagramSocket server = new DatagramSocket(PUERTO);
            InetAddress ip = InetAddress.getByName("localhost");
            // Crea el espacio para los mensajes
            byte[] recibe = new byte[1000];
            DatagramPacket mensaje = new DatagramPacket(recibe, recibe.length);
            boolean fin = false;
            System.out.println("Servidor esperando mensajes en el puerto: "+PUERTO);
            do{
                // Recibe el mensaje
                server.receive(mensaje);
                // convierte el mensaje de byte a String
                String datos = new String(mensaje.getData(), 0, mensaje.getLength());
                String respuesta = "";
                if(datos.equalsIgnoreCase("como te llamas?")){
```

```
                    respuesta = "Me llamo Izan";
                }else if(datos.equalsIgnoreCase("cual es tu edad?")){
                    respuesta = "Tengo 30 años";
                }else if(datos.equalsIgnoreCase("fin")){
                    respuesta = "Fin de la conversación";
                }else{
                    respuesta = "No he entendido la pregunta";
                }
                // envia la respuesta convertida a byte
                byte[] enviar = respuesta.getBytes();
                // crea el paquete con la respuesta
                DatagramPacket respuestaPaquete = new DatagramPacket(enviar, enviar.length, ip,
mensaje.getPort());
                // envia la respuesta
                server.send(respuestaPaquete);
            }while(!fin);
            // cuando se adivina el numero secreto se cierra el socket y se termina el programa
            server.close();
        }catch (Exception e){
            e.printStackTrace();
        }
    }
}
```

CON HILOS

```
public class Servidor extends Thread{
    static final int PUERTO = 2000;
    private DatagramSocket server;

    public Servidor(DatagramPacket mensaje, DatagramSocket server, InetAddress ip){
        try{
            boolean fin = false;
            while(!fin){
                // Recibe el mensaje
                server.receive(mensaje);
                // convierte el mensaje de byte a String
                String datos = new String(mensaje.getData(), 0, mensaje.getLength());
                String respuesta = "";
                if(datos.equalsIgnoreCase("como te llamas?")){
                    respuesta = "Me llamo Izan";
                }else if(datos.equalsIgnoreCase("cual es tu edad?")){
                    respuesta = "Tengo 30 años";
                }else if(datos.equalsIgnoreCase("fin")){
                    respuesta = "Fin de la conversación";
                }else{
                    respuesta = "No he entendido la pregunta";
                }
                // envia la respuesta convertida a byte
                byte[] enviar = respuesta.getBytes();
                // crea el paquete con la respuesta
                DatagramPacket respuestaPaquete = new DatagramPacket(enviar, enviar.length, ip,
mensaje.getPort());
                // envia la respuesta
                server.send(respuestaPaquete);
            }
        } catch (Exception e){
            e.printStackTrace();
        }
    }
    public static void main(String [] args){
        try{
            // Se crea el socket del servidor
            DatagramSocket server = new DatagramSocket(PUERTO);
            InetAddress ip = InetAddress.getByName("localhost");
            // Crea el espacio para los mensajes
            byte[] recibe = new byte[1000];
            DatagramPacket mensaje = new DatagramPacket(recibe, recibe.length);
            System.out.println("Servidor esperando mensajes en el puerto: "+PUERTO);
            new Servidor(mensaje, server, ip).start();
        }catch (Exception e){
            e.printStackTrace();
        }
```

```
    }
}
```

```
public class Cliente {
    static final int PUERTO = 2000;

    public static void main(String [] args){
        String pregunta;
        Scanner t = new Scanner(System.in);
        DatagramSocket cliente = null;
        do {
            System.out.print("Realizar pregunta:");
            pregunta = t.nextLine();
            try{
                // Crear el socket
                cliente = new DatagramSocket();
                // construye la dirección del receptor pasando por argumento la máquina destino
                InetAddress maquina = InetAddress.getByName("localhost");
                // construye el mensaje
                byte[] cadena = pregunta.getBytes();
                DatagramPacket mensaje = new DatagramPacket(cadena, pregunta.length(), maquina,
PUERTO);
                // envía el mensaje
                cliente.send(mensaje);
                // Crea el espacio para los mensajes
                byte[] recibe = new byte[1000];
                //  crea el paquete con la respuesta
                DatagramPacket respuesta = new DatagramPacket(recibe, recibe.length);
                // Recibe el mensaje
                cliente.receive(respuesta);
                // convierte el mensaje de byte a String
                String datos = new String(respuesta.getData(), 0, respuesta.getLength());
                System.out.println(datos);
            }catch(Exception e){
                e.printStackTrace();
            }
        }while(!pregunta.equalsIgnoreCase("fin"));
        // cierra el socket
        cliente.close();
    }
}
```

```
El objetivo del ejercicio es crear una aplicación cliente/servidor usando sockets
datagram que permita el envío de objetos de clase Persona entre los equipos. El
cliente enviará un objeto (serializable) al servidor y este mostrará por pantalla los
datos recibidos, asignará un valor al campo registro del objeto y lo devolverá al
cliente para que reescriba el objeto enviado y mostrará por pantalla los datos.
La clase Persona contendrá los datos: nombre, edad, DNI y registro.
```

```
public class Persona implements Serializable {
    private String nombre;
    private int edad;
    private String DNI;
    private int registro;

    public Persona(String nombre, int edad, String DNI) {
        this.nombre = nombre;
        this.edad = edad;
        this.DNI = DNI;
    }
    public String getNombre() {
        return nombre;
    }
    public int getEdad() {
        return edad;
    }
    public String getDNI() {
        return DNI;
    }
```

```
    public int getRegistro() {
        return registro;
    }
    public void setRegistro(int registro) {
        this.registro = registro;
    }
    @Override
    public String toString() {
        return nombre+" "+edad+" "+DNI+" "+registro;
    }
}
```

```
public static void main(String [] args){
        try{
            // Se crea el socket del servidor
            DatagramSocket server = new DatagramSocket(PUERTO);
            // Crea el espacio para los mensajes
            byte[] recibe = new byte[1000];
            byte[] envia;
            DatagramPacket mensaje = new DatagramPacket(recibe, recibe.length);
            System.out.println("Servidor esperando mensajes en el puerto: "+PUERTO);
            while (true) {
                // Recibe el mensaje
                DatagramPacket llegada = new DatagramPacket(recibe, recibe.length);
                server.receive(llegada);
                // convierte el mensaje de byte a Persona
                ByteArrayInputStream bais = new ByteArrayInputStream(llegada.getData());
                ObjectInputStream ois = new ObjectInputStream(bais);
                // Deserializamos el objeto Persona
                Persona p = (Persona) ois.readObject();
                System.out.println("Recibido: "+p.toString());
                p.setRegistro(registro++);
                // Creamos un objeto para serializar el objeto Persona en un array de bytes
                ByteArrayOutputStream baos = new ByteArrayOutputStream();
                ObjectOutputStream oos = new ObjectOutputStream(baos);
                // Serializamos el objeto Persona
                oos.writeObject(p);
                envia = baos.toByteArray();
                // crea el paquete con la respuesta
                DatagramPacket respuesta = new DatagramPacket(envia, envia.length,
llegada.getAddress(), llegada.getPort());
                server.send(respuesta);
                System.out.println("Enviado: "+p.toString());
            }
        }catch (Exception e){
            e.printStackTrace();
        }
    }
}
```

CON HILOS

```
public class Servidor extends Thread {
    static final int PUERTO = 2000;
    private static int registro = 1;
    private byte[] recibe;
    private byte[] envia;
    private DatagramSocket server;

    public Servidor(byte[] recibe, byte[] envia, DatagramSocket server){
        this.recibe = recibe;
        this.envia = envia;
        this.server = server;
    }
    public void run(){
        try{
            while (true) {
                // Recibe el mensaje
                DatagramPacket llegada = new DatagramPacket(recibe, recibe.length);
                server.receive(llegada);
                // convierte el mensaje de byte a Persona
                ByteArrayInputStream bais = new ByteArrayInputStream(llegada.getData());
```

```
                ObjectInputStream ois = new ObjectInputStream(bais);
                // Deserializamos el objeto Persona
                Persona p = (Persona) ois.readObject();
                System.out.println("Recibido: "+p.toString());
                p.setRegistro(registro++);
                // Creamos un objeto para serializar el objeto Persona en un array de bytes
                ByteArrayOutputStream baos = new ByteArrayOutputStream();
                ObjectOutputStream oos = new ObjectOutputStream(baos);
                // Serializamos el objeto Persona
                oos.writeObject(p);
                envia = baos.toByteArray();
                // crea el paquete con la respuesta
                DatagramPacket respuesta = new DatagramPacket(envia, envia.length,
llegada.getAddress(), llegada.getPort());
                server.send(respuesta);
                System.out.println("Enviado: "+p.toString());
            }
        }
        catch (Exception e) {
            e.printStackTrace();
        }
    }
    public static void main(String [] args){
        try{
            // Se crea el socket del servidor
            DatagramSocket server = new DatagramSocket(PUERTO);
            // Crea el espacio para los mensajes
            byte[] recibe = new byte[1000];
            byte[] envia = null;
            DatagramPacket mensaje = new DatagramPacket(recibe, recibe.length);
            System.out.println("Servidor esperando mensajes en el puerto: "+PUERTO);
            new Servidor(recibe, envia, server).start();
        }catch (Exception e){
            e.printStackTrace();
        }
    }
}
```

```
public class Cliente {
    static final int PUERTO = 2000;

    public static void main(String[] args) {
        Scanner t = new Scanner(System.in);
        DatagramSocket cliente = null;
        System.out.print("Nombre:");
        String nombre = t.nextLine();
        System.out.print("Edad:");
        int edad = Integer.parseInt(t.nextLine());
        System.out.print("DNI:");
        String DNI = t.nextLine();
        Persona p = new Persona(nombre, edad, DNI);
        p.setRegistro(0);
        try {
            // Crear el socket
            cliente = new DatagramSocket();
            // construye la dirección del receptor pasando por argumento la máquina destino
            InetAddress maquina = InetAddress.getByName("localhost");
            // Crea el espacio para los mensajes
            byte[] envia;
            // Crea el espacio para los mensajes
            byte[] recibe = new byte[1000];
            // Creamos un objeto para serializar el objeto Persona en un array de bytes
            ByteArrayOutputStream baos = new ByteArrayOutputStream();
            ObjectOutputStream oos = new ObjectOutputStream(baos);
            // Serializamos el objeto Persona
            oos.writeObject(p);
            envia = baos.toByteArray();
            // construye el mensaje
            DatagramPacket mensaje = new DatagramPacket(envia, envia.length, maquina, PUERTO);
            // envía el mensaje
            cliente.send(mensaje);
```

```
            //  crea el paquete con la respuesta
            DatagramPacket respuesta = new DatagramPacket(recibe, recibe.length);
            // Recibe el mensaje
            cliente.receive(respuesta);
            // Creamos un objeto para deserializar el array de Bytes en un objeto Persona
            ByteArrayInputStream bais = new ByteArrayInputStream(respuesta.getData());
            ObjectInputStream ois = new ObjectInputStream(bais);
            // Deserializamos el objeto Persona
            p = (Persona) ois.readObject();
            System.out.println(p.toString());
        } catch (Exception e) {
            e.printStackTrace();
        }
        cliente.close();// cierra el socket
    }
}
```

OTRA FORMA DE ENVIAR OBJETOS

```
public class Persona implements Serializable {
    private String nombre;
    private int edad;
    public Persona (String nombre, int edad){
        super();
        this.nombre=nombre;
        this.edad=edad;
    }
    public String getNombre (){
        return nombre;
    }
    public void setNombre(String n){
        this.nombre=n;
    }
    public int getEdad (){
        return edad;
    }
    public void setEdad (int a){
        this.edad=a;
    }
}
```

```
public class Servidor {
    static final int Puerto = 1500;
    public Servidor() {
        try {
            // Inicio el servidor en el puerto
            ServerSocket skServidor = new ServerSocket(Puerto);
            System.out.println("Escucho el puerto " + Puerto);
            Socket skCliente = skServidor.accept(); // Crea objeto
            // Creo el flujo de salida
            ObjectOutputStream flujo_salida = new
ObjectOutputStream(skCliente.getOutputStream());
            // Creo el flujo de entrada
            ObjectInputStream flujo_entrada = new ObjectInputStream(skCliente.getInputStream());
            System.out.println("Cliente conectado");
            Persona persona = new Persona("Pablo", 35);
            flujo_salida.writeObject(persona);
            // Se cierra la conexión
            skCliente.close();
            flujo_entrada.close();
            flujo_salida.close();
            System.out.println("Cliente desconectado");
        } catch (IOException ex) {
            ex.getMessage();
        }
    }
    public static void main(String[] args) {
        new Servidor();
    }
}
public class Cliente {
    static final String HOST = "localhost";
```

```
    static final int Puerto = 1500;
    public Cliente() {
        Persona persona = null;
        try {
            // Me conecto al puerto
            Socket skCliente = new Socket(HOST, Puerto);
            // Creo el flujo de salida
            ObjectOutputStream flujo_salida = new
ObjectOutputStream(skCliente.getOutputStream());
            // Creo el flujo de entrada
            ObjectInputStream flujo_entrada = new ObjectInputStream(skCliente.getInputStream());
            System.out.println("Conectado al servidor");
            persona = (Persona) flujo_entrada.readObject();
            System.out.println("Recibido: " + persona.getNombre() + " " + persona.getEdad());
            // Se cierra la conexión
            skCliente.close();
            flujo_entrada.close();
            flujo_salida.close();
            System.out.println("Cliente desconectado");
        } catch (Exception e) {
            System.err.println(e.getMessage() + " " + e.toString());
        }
    }
    public static void main(String[] args) {
        new Cliente();
    }
}
```