1. ServidorUDP.java (Multijugador)

Este servidor utiliza un HashMap para guardar el número secreto de cada cliente de forma independiente.

Java

```
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.util.HashMap;

public class ServidorUDP {
    private static final int PUERTO = 2000;
    // Mapa para guardar: "IP:Puerto" -> NumeroSecreto
    private static HashMap<String, Integer> partidas = new
HashMap<>();

    public static void main(String[] args) {
        try (DatagramSocket socket = new DatagramSocket(PUERTO)) {
            System.out.println("Servidor UDP iniciado. Esperando
paquetes...");

            byte[] bufferEntrada = new byte[1024];

            while (true) {
                // 1. Recibir paquete
                DatagramPacket paqueteRecibido = new
DatagramPacket(bufferEntrada, bufferEntrada.length);
                socket.receive(paqueteRecibido);

                String mensaje = new
String(paqueteRecibido.getData(), 0,
paqueteRecibido.getLength()).trim();
                InetAddress ipCliente =
paqueteRecibido.getAddress();
                int puertoCliente = paqueteRecibido.getPort();

                // Identificador único para este cliente
                String idCliente = ipCliente.getHostAddress() + ":"
+ puertoCliente;
```

```
                String respuesta;

                // Lógica del juego
                if (mensaje.equalsIgnoreCase("HOLA")) {
                    int numS = (int) (Math.random() * 10);
                    partidas.put(idCliente, numS); // Guardamos su
número secreto
                    respuesta = "Bienvenido! Adivina del 0 al 9.
(Nuevo numero generado)";
                    System.out.println("Nueva partida para: " +
idCliente + " (Num: " + numS + ")");
                } else {
                    try {
                        int intento = Integer.parseInt(mensaje);
                        Integer numSecreto =
partidas.get(idCliente);

                        if (numSecreto == null) {
                            respuesta = "Error: Primero debes enviar
'HOLA' para empezar.";
                        } else if (intento == numSecreto) {
                            respuesta = "Has ganado!";
                            partidas.remove(idCliente); // Borramos
partida al ganar
                        } else if (intento < numSecreto) {
                            respuesta = "El numero es mayor";
                        } else {
                            respuesta = "El numero es menor";
                        }
                    } catch (NumberFormatException e) {
                        respuesta = "Envia un numero valido.";
                    }
                }

                // 2. Enviar respuesta
                byte[] datosRespuesta = respuesta.getBytes();
                DatagramPacket paqueteEnviar = new DatagramPacket(
                    datosRespuesta, datosRespuesta.length,
ipCliente, puertoCliente
                );
                socket.send(paqueteEnviar);
```

```
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

2. ClienteUDP.java

El cliente es más sencillo, pero fíjate cómo manejamos el envío y recepción de paquetes en lugar de usar Streams.

Java

```
import java.io.IOException;
import java.net.*;
import java.util.Scanner;

public class ClienteUDP {
    private static final int PUERTO = 2000;
    private static final String HOST = "localhost";

    public static void main(String[] args) {
        try (DatagramSocket socket = new DatagramSocket();
             Scanner sc = new Scanner(System.in)) {

            InetAddress ipServidor = InetAddress.getByName(HOST);
            byte[] bufferRecibo = new byte[1024];

            // 1. Iniciar conexión (HOLA)
            enviar(socket, "HOLA", ipServidor, PUERTO);

            boolean juegoVivo = true;
            while (juegoVivo) {
                // 2. Recibir respuesta del servidor
                DatagramPacket recibo = new
DatagramPacket(bufferRecibo, bufferRecibo.length);
                socket.receive(recibo);
                String msgServidor = new String(recibo.getData(), 0,
recibo.getLength());
                System.out.println("SERVIDOR: " + msgServidor);
```

```
                if (msgServidor.contains("ganado")) break;

                // 3. Leer entrada de usuario y enviar
                System.out.print("Tu numero: ");
                if (sc.hasNextInt()) {
                    String intento = String.valueOf(sc.nextInt());
                    sc.nextLine();
                    enviar(socket, intento, ipServidor, PUERTO);
                } else {
                    sc.next();
                    System.out.println("Introduce un numero
entero.");
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private static void enviar(DatagramSocket socket, String msj,
InetAddress ip, int puerto) throws IOException {
        byte[] b = msj.getBytes();
        DatagramPacket p = new DatagramPacket(b, b.length, ip,
puerto);
        socket.send(p);
    }
}
```

Resumen de diferencias que verás al ejecutarlo:

- **Identificación:** En el Servidor UDP, usamos `ipCliente.getHostAddress() + ":" + puertoCliente`. Esto es necesario porque **todos** los clientes entran por el mismo puerto 2000 del servidor. Sin esto, el servidor no sabría si el "5" que le llega es del Jugador A o del Jugador B.
- **Conexión:** Si cierras el cliente y lo vuelves a abrir, el servidor te tratará como alguien nuevo porque tu puerto de salida (puerto efímero) habrá cambiado.
- **Sin `dis.readUTF()`**: Al usar `new String(paquete.getData())`, estamos convirtiendo los bytes crudos a texto. Es más manual pero más rápido.

SIMPLE

En esta versión, el numeroSecreto es una variable simple.

Java

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

public class ServidorUDPSimple {
    public static void main(String[] args) {
        try (DatagramSocket socket = new DatagramSocket(2000)) {
            System.out.println("Servidor UDP Simple (1 solo
jugador)");

            // El número es el mismo para el que llegue primero
            int numeroSecreto = (int) (Math.random() * 10);
            byte[] buffer = new byte[1024];

            while (true) {
                // Recibir el número del cliente
                DatagramPacket pedido = new DatagramPacket(buffer,
buffer.length);
                socket.receive(pedido);

                String mensaje = new String(pedido.getData(), 0,
pedido.getLength()).trim();
                int intento = Integer.parseInt(mensaje);

                String respuesta;
                if (intento == numeroSecreto) {
                    respuesta = "¡Ganaste! Generando nuevo
numero...";
                    numeroSecreto = (int) (Math.random() * 10); //
Reiniciamos para el siguiente
                } else {
                    respuesta = (intento < numeroSecreto) ? "Es
mayor" : "Es menor";
                }

                // Enviar respuesta a quien sea que haya enviado el
```

```
paquete
                byte[] dataRes = respuesta.getBytes();
                DatagramPacket respuestaPaq = new DatagramPacket(
                    dataRes, dataRes.length, pedido.getAddress(),
pedido.getPort()
                );
                socket.send(respuestaPaq);
            }
        } catch (IOException e) { e.printStackTrace(); }
    }
}
```

2. Cliente UDP Simplificado

Este cliente es directo: envía un número y espera respuesta.

Java

```
import java.io.IOException;
import java.net.*;
import java.util.Scanner;

public class ClienteUDPSimple {
    public static void main(String[] args) {
        try (DatagramSocket socket = new DatagramSocket();
             Scanner sc = new Scanner(System.in)) {

            InetAddress ip = InetAddress.getByName("localhost");

            while (true) {
                System.out.print("Adivina el numero: ");
                String num = sc.nextLine();

                // Enviar
                byte[] b = num.getBytes();
                socket.send(new DatagramPacket(b, b.length, ip, 200
```