

Tarea PSP03

Actividad 3.1: Juego "Adivina el Número" (Sockets TCP)

1. Descripción de la solución

En esta actividad he creado una conexión TCP en el puerto 2000. El servidor genera un número aleatorio del 0 al 100 y utiliza un bucle para recibir intentos del cliente, respondiendo con pistas hasta que este acierta.

2. Código Fuente

Clase: ServidorAdivina.java

```
package actividad31;

import java.io.*;
import java.net.*;

public class ServidorAdivina {
    static final int PUERTO = 2000;

    public static void main(String[] args) {
        try {
            ServerSocket skServidor = new ServerSocket(PUERTO);
            System.out.println("Servidor Adivina escuchando en el
puerto " + PUERTO);

            // Esperar cliente
            Socket sCliente = skServidor.accept();
            System.out.println("Cliente conectado para jugar.");

            // Flujos
            DataOutputStream flujo_salida = new
DataOutputStream(sCliente.getOutputStream());
            DataInputStream flujo_entrada = new
DataInputStream(sCliente.getInputStream());
```

```

// Generar número secreto (0 al 100)
int numeroSecreto = (int) (Math.random() * 101);

boolean juegoTerminado = false;

while (!juegoTerminado) {
    // Leer intento del cliente
    int numeroCliente = flujo_entrada.readInt();

    if (numeroCliente == numeroSecreto) {
        flujo_salida.writeUTF("Acertaste");
        juegoTerminado = true;
    } else if (numeroCliente < numeroSecreto) {
        flujo_salida.writeUTF("Mayor");
    } else {
        flujo_salida.writeUTF("Menor");
    }
}

System.out.println("El cliente ha acertado. Cerrando
servidor.");
sCliente.close();
skServidor.close();

} catch (Exception e) {
    System.out.println("Error en servidor: " +
e.getMessage());
}
}
}

```

Clase: ClienteAdivina.java

```

package actividadad31;

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

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

```

```
public static void main(String[] args) {
    try {
        Socket sCliente = new Socket(HOST, PUERTO);
        System.out.println("Conectado al servidor. ¡Adivina el
número del 0 al 100!");

        DataInputStream flujo_entrada = new
DataInputStream(sCliente.getInputStream());
        DataOutputStream flujo_salida = new
DataOutputStream(sCliente.getOutputStream());

        Scanner sc = new Scanner(System.in);
        boolean acertado = false;

        while (!acertado) {
            System.out.print("Introduce tu número: ");
            int miNumero = sc.nextInt();

            // Enviar número
            flujo_salida.writeInt(miNumero);

            // Recibir respuesta
            String respuesta = flujo_entrada.readUTF();
            System.out.println("Servidor dice: " + respuesta);

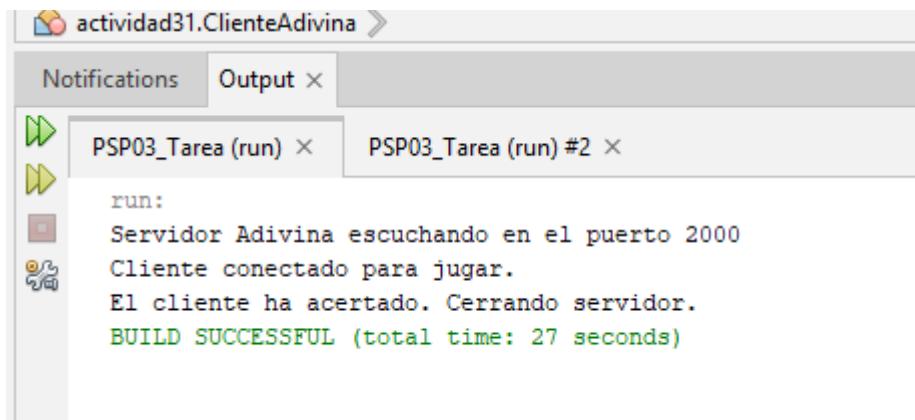
            if (respuesta.equals("Acertaste")) {
                acertado = true;
                System.out.println("¡Juego terminado!");
            }
        }

        sCliente.close();

    } catch (Exception e) {
        System.out.println("Error en cliente: " +
e.getMessage());
    }
}
```

3. Capturas de pantalla de la ejecución

1. La consola del Servidor



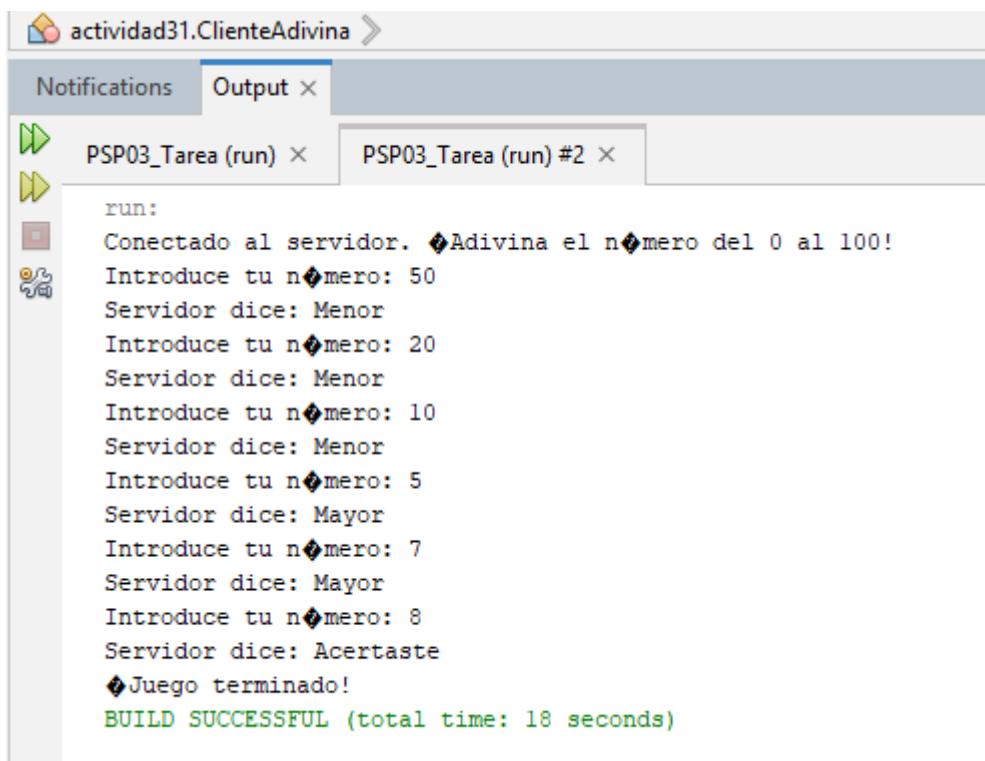
The screenshot shows the Eclipse IDE's Output view for the project "actividad31.ClienteAdivina". There are two tabs: "Notifications" and "Output". The "Output" tab is selected and contains two entries:

- PSP03_Tarea (run) ×
- PSP03_Tarea (run) #2 ×

Content of the log:

```
run:  
Servidor Adivina escuchando en el puerto 2000  
Cliente conectado para jugar.  
El cliente ha acertado. Cerrando servidor.  
BUILD SUCCESSFUL (total time: 27 seconds)
```

2. La consola del Cliente



The screenshot shows the Eclipse IDE's Output view for the project "actividad31.ClienteAdivina". There are two tabs: "Notifications" and "Output". The "Output" tab is selected and contains two entries:

- PSP03_Tarea (run) ×
- PSP03_Tarea (run) #2 ×

Content of the log:

```
run:  
Conectado al servidor. ♦Adivina el n♦mero del 0 al 100!  
Introduce tu n♦mero: 50  
Servidor dice: Menor  
Introduce tu n♦mero: 20  
Servidor dice: Menor  
Introduce tu n♦mero: 10  
Servidor dice: Menor  
Introduce tu n♦mero: 5  
Servidor dice: Mayor  
Introduce tu n♦mero: 7  
Servidor dice: Mayor  
Introduce tu n♦mero: 8  
Servidor dice: Acertaste  
♦Juego terminado!  
BUILD SUCCESSFUL (total time: 18 seconds)
```

Actividad 3.2: Servidor de Ficheros

1. Descripción de la solución

He implementado un sistema de transferencia de archivos por el puerto 1500. El cliente envía una cadena de texto con el nombre del archivo y el servidor verifica su existencia usando la clase File. Si no existe un archivo con el nombre especificado se rechaza la

petición, y si existe (en la raíz del proyecto) se envía el contenido del archivo al cliente y este lo lee por pantalla. Después se cierra.

2. Código Fuente

Clase: ServidorFichero.java

```
package actividad32;

import java.io.*;
import java.net.*;

public class ServidorFichero {
    static final int PUERTO = 1500;

    public static void main(String[] args) {
        try {
            ServerSocket skServidor = new ServerSocket(PUERTO);
            System.out.println("Servidor de Ficheros listo en puerto " + PUERTO);

            Socket sCliente = skServidor.accept();
            System.out.println("Cliente conectado.");

            DataInputStream flujo_entrada = new
DataInputStream(sCliente.getInputStream());
            DataOutputStream flujo_salida = new
DataOutputStream(sCliente.getOutputStream());

            boolean ficheroEnviado = false;

            // BUCLE: Repetimos mientras NO hayamos enviado el fichero
            while (!ficheroEnviado) {
                // 1. Recibir nombre del fichero
                String nombreFichero = flujo_entrada.readUTF();
                File fichero = new File(nombreFichero);

                // 2. Comprobar existencia
                if (fichero.exists() && fichero.isFile()) {
                    flujo_salida.writeUTF("OK"); // Avisamos al cliente que existe

                    // Leemos y enviamos el fichero
                    byte[] buffer = new byte[1024];
                    int bytesLeidos;
                    while ((bytesLeidos = flujo_entrada.read(buffer)) > 0) {
                        flujo_salida.write(buffer, 0, bytesLeidos);
                    }
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

```

        BufferedReader lector = new BufferedReader(new FileReader(fichero));
        String linea;
        while ((linea = lector.readLine()) != null) {
            flujo_salida.writeUTF(linea);
        }
        lector.close();

        // Señal de fin de fichero
        flujo_salida.writeUTF("EOF");

        // Marcamos como enviado para salir del bucle
        ficheroEnviado = true;
        System.out.println("Fichero enviado con éxito. Cerrando conexión.");

    } else {
        flujo_salida.writeUTF("ERROR");
        // Volvemos al inicio del bucle a esperar otro nombre
    }
}

sCliente.close();
skServidor.close();

} catch (Exception e) {
    System.out.println("Error o cliente desconectado: " + e.getMessage());
}
}
}

```

Clase: ClienteFichero.java

```

package actividad32;

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

public class ClienteFichero {
    static final String HOST = "localhost";
    static final int PUERTO = 1500;

```

```
public static void main(String[] args) {
    try {
        Scanner sc = new Scanner(System.in);
        Socket sCliente = new Socket(HOST, PUERTO);

        DataOutputStream flujo_salida = new
DataOutputStream(sCliente.getOutputStream());
        DataInputStream flujo_entrada = new
DataInputStream(sCliente.getInputStream());

        boolean ficheroRecibido = false;

        // Repetimos hasta recibir el fichero correctamente
        while (!ficheroRecibido) {
            System.out.print("Introduce el nombre del fichero que quieras ver: ");
            String ficheroSolicitado = sc.nextLine();

            flujo_salida.writeUTF(ficheroSolicitado);

            String estado = flujo_entrada.readUTF();

            if (estado.equals("OK")) {
                System.out.println("--- Contenido del Fichero ---");

                String linea = flujo_entrada.readUTF();
                while (!linea.equals("EOF")) {
                    System.out.println(linea);
                    linea = flujo_entrada.readUTF();
                }
                System.out.println("-----");
                System.out.println("Fichero recibido. Finalizando conexión.");
                ficheroRecibido = true; // Salimos del bucle
            } else if (estado.equals("ERROR")) {
                System.out.println("Error: El fichero no existe. Inténtalo de nuevo.\n");
            }
        }

        sCliente.close();
    }
```

```

        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}

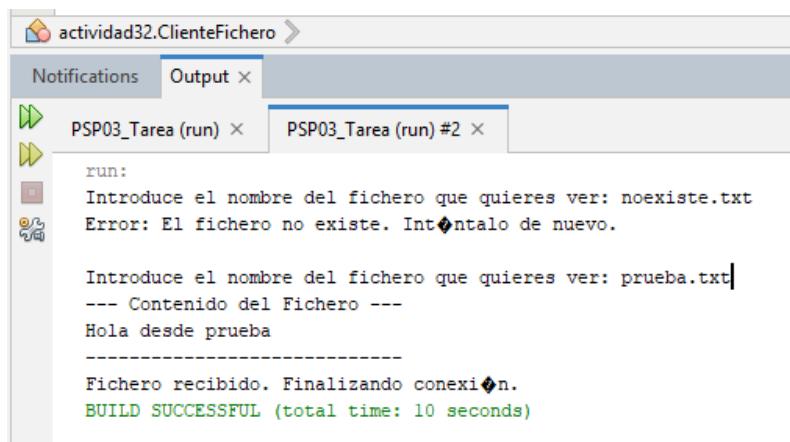
```

3. Pruebas de Ejecución (Capturas de Pantalla)

Prueba 1: El fichero NO existe

Prueba 2: El fichero existe

Parte del Cliente



The screenshot shows a terminal window titled "actividad32.ClienteFichero". It has two tabs: "Notifications" and "Output". The "Output" tab is selected. There are two entries in the list:

- PSP03_Tarea (run) ×
- PSP03_Tarea (run) #2 ×

The content of the window is as follows:

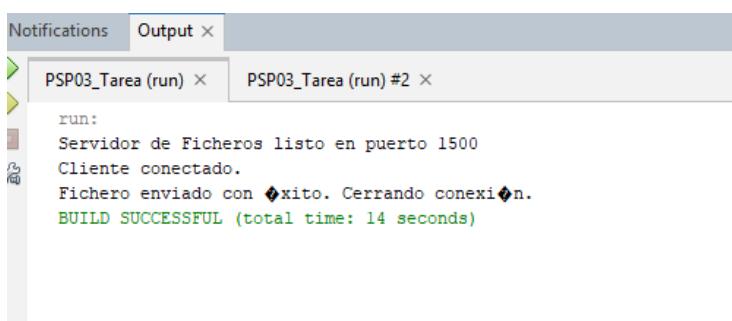
```

run:
Introduce el nombre del fichero que quieras ver: noexiste.txt
Error: El fichero no existe. Intentalo de nuevo.

Introduce el nombre del fichero que quieras ver: prueba.txt
--- Contenido del Fichero ---
Hola desde prueba
-----
Fichero recibido. Finalizando conexión.
BUILD SUCCESSFUL (total time: 10 seconds)

```

Parte del servidor



The screenshot shows a terminal window with the same title and tabs as the client window. It has two entries in the list:

- PSP03_Tarea (run) ×
- PSP03_Tarea (run) #2 ×

The content of the window is as follows:

```

run:
Servidor de Ficheros listo en puerto 1500
Cliente conectado.
Fichero enviado con éxito. Cerrando conexión.
BUILD SUCCESSFUL (total time: 14 seconds)

```