3.8. CONEXION DE MULTIPLES CLIENTES. HILOS

Hasta ahora los programas servidores que hemos creado solo son capaces de atender a un cliente en cada momento, pero lo más típico es que un programa servidor pueda atender a muchos clientes simultáneamente. La solución para poder atender a múltiples clientes está en recurrir a procesos multihilo, cada cliente será atendido en un hilo.

El esquema básico en sockets TCP seria construir un único servidor con la clase ServerSocket e invocar al método accept() para esperar las peticiones de conexión de los clientes. Cuando un cliente se conecta, el método accept() devuelve un objeto Socket, éste se usara para crear un hilo cuya misión es atender a este cliente. Después se vuelve a invocar a accept() para esperar a un nuevo cliente; habitualmente la espera de conexiones se hace dentro de un bucle infinito:

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

public class Servidor {
public static void main(String args[]) throws IOException {
ServerSocket servidor;
servidor = new ServerSocket (6000) ;
System.out.println("Servidor iniciado...");

while (true) {
Socket cliente = new Sccket() ;
cliente = servidor.accept();//esperando cliente
HiloServidor hilo = new HiloServidor (cliente) ;
hilo.start(); //se atiende al cliente
}
}
}//..
```

Todas las operaciones que sirven a un cliente en particular quedan dentro de la clase hilo. El hilo permite que el servidor se mantenga a la escucha de peticiones y no interrumpa su proceso mientras los clientes son atendidos.

Por ejemplo, supongamos que el cliente envía una cadena de caracteres al servidor y el servidor se la devuelve en mayúsculas, hasta que recibe un asterisco que finaliza la comunicación con el cliente (por claridad se han eliminado los bloques try-catch). El proceso de tratamiento de la cadena se realiza en un hilo, en este caso se llama HiloServidor:

```
import java.io,*;
import java.net.*;

public class HiloServidor extends Thread {
BufferedReader fentrada;
PrintWriter fsalida;
Socket socket = null;

public HiloServidor (Socket s) {//CONSTRUCTOR
socket = s;
//SE CREAN FLUJOS DE ENTRADA Y SALIDA CON EBL CLIENTE
fsalida = new PrintWriter (socket.getOutputStream(), true);
fentrada = new BufferedReader (
new InputStreamReader (socket.getInputStream()));
}
public void run() {//tarea a realizar con el cliente
String cadena = "";
System.out.println("COMUNICO CON: "+ socket.toString()};
while (!cadena.trim() .equals("*")) {
cadena = fentrada.readLine() ;//obtener cadena
fsalida.printin (cadena.trim() .toUpperCase());//enviar mayuscula
}// fin while

System.out.println("FIN CON: "+ socket.toString());

fsalida.close();
fentrada.close();
socket.close();

}//fin run

}//..
```

Como programa cliente podemos ejecutar el programa Cliente.java que se conectara con el servidor en el puerto 6000 y le envían cadenas introducidas por teclado; cuando le envíe un asterisco el servidor finalizara la comunicación con el cliente:

```
import java.io.*;
import java.net.*;
        public class Cliente {
                public static void main(String[] args) throws IOException {
                        String Host = "localhost";
                        int Puerto = 6000;// puerto remoto
                        Socket Cliente = new Socket (Host, Puerto) ;
                        //SE CREAN LOS FLUJOS DE ENTRADA Y SALIDA
                        PrintWriter fsalida = new PrintWriter
                                        (Cliente.getOutputStream(), true);
                        BufferedReader fentrada = new BufferedReader
                                (new InputStreamReader (Cliente.getInputStream()));
        //FLUJO PARA ENTRADA ESTANDAR
        BufferedReader in = new
                BufferedReader (new InputStreamReader (System.in));
        String cadena, eco = "";
        do {
                System.out.print ("Introduce cadena: ");
                cadena = in.readLine();
                fsalida.println(cadena) ; //envio cadena al servidor
                eco = fentrada.readLine(); //recibo cadena del servidor
                System.out.printin(" =>ECO: " + eco);
                } while (!cadena.trim().equals("*"));
                fsalida.close();
                fentrada.close();
                System.out.println("Fin del envio... ");
                in.close();
                Cliente.close();
                }//
}//..
```

ACTIVIDAD 3.8

Realiza un programa servidor que escuche en el puerto 44444. Cada vez que se conecte un cliente se creará un nuevo hilo para atenderlo. Se mostrará en la consola del servidor la dirección IP y el puerto remoto del cliente que se conecta y cuando el cliente se desconecte se deberá mostrar un mensaje indicando que se ha desconectado.