PRACTICA : CREACIÓN DE UN CHAT CON TCP

Un ejemplo típico de un servidor que atiende a múltiples clientes es un servidor de chat.

En esta practica, cada cliente será atendido en un hilo de ejecución; en ese hilo se recibirán sus mensajes y se enviarán al resto de miembros del chat.

Clase servidorChat:
El programa servidor va ha definir el número máximo de conexiones que admite e irá controlando los clientes que actualmente estén conectados, para ello utiliza un objeto de la clase "**ComunHilos**" que será compartido por todos los hilos .

Atributos del objeto ComunHilos:

- ***int conexiones*** : Almacena el número de conexiones de clientes. Cada vez que se conecta un cliente sumamos 1 a este atributo y lo usamos como índice para ir llenando el array de sockets con los clientes que se van conectando. El máximo de conexiones permitidas lo indica el atributo "maximo".
- ***int actuales:*** Almacena el número de clientes conectados en este momento. Cada vez que se desconecta un cliente se resta 1 a este atributo.
- ***int maximo***: Atributo que indica el número máximo de clientes que se pueden conectar.
- ***Socket tabla[] = new Socket[maximo]***: Array que almacena los sockets de los clientes que se conectan. Usaremos el array para tener control de los clientes y así poder enviarles la conversación del chat cada vez que uno envía algún mensaje.
- ***String mensaje***: Contiene los mensajes del chat.

El programa servidor deberá realizar las siguientes tareas:
- Definir una variable con el máximo número de conexiones permitidas (por ejemplo 5)
- Crear el ServerSocket.
- Crear un array para llevar el control de los clientes conectados.
- Crear un objeto de tipo "ComunHilos" donde se inicializan todas las variables comentadas anteriormente.
- Debe de realizar un bucle para controlar el número de conexiones.
- Dentro del bucle el servidor espera la conexión del cliente y cuando se conecta se crea un socket.
- El socket creado se almacenará en el array, se incrementa el número de conexiones y las conexiones actuales.
- Lanzará el hilo para gestionar los mensajes del cliente que se acaba de conectar.

```
public class ServidorChat
```

Atributo

```
static final int MAXIMO = 5;//MAXIMO DE CONEXIONES PERMITIDAS
```

public static void main(String args[]) throws IOException

Clase HiloServidorChat

Al lanzar el hilo se envía al constructor el socket creado y el objeto "ComunHilos" compartido por todos los hilos. Al llegar al máximo de conexiones se cierra el "***ServerSocket***" y finaliza el proceso servidor, los clientes que estaban conectados seguirán funcionando.

El hilo "***HiloServidorChat***" se encarga de recibir y enviar los mensajes a los clientes de chat. En el constructor, se recibe el socket creado y el objeto compartido por todos los hilos. Se crea el flujo de entrada desde el que se leen los mensajes que el cliente de chat envía:

En el método "run()", lo primero que hacemos es enviar los mensajes que hay actualmente en el chat al programa cliente para que los muestre por pantalla. Esto se hace en el método "EnviarMensajesaTodos()". Los mensajes que se envían son los que en ese momento hay en el chat.

A continuación, se hace un bucle while en el que se recibe lo que el cliente escribe en el chat. Cuando un cliente finaliza (pulsa el botón Salir de su pantalla) envía un asterisco al servidor de chat, entonces se sale del bucle while, ya que termina el proceso del cliente, de esta manera se controlan las conexiones actuales.

El texto que el cliente escribe en su chat, se añade al atributo "mensajes" del objeto compartido para poder enviar la conversación a todos los clientes, el método "EnviarMensajesaTodos()" se encargará de ello. Después del bucle while se cierra el socket del cliente.

El método "***EnviarMensajesaTodos()***" envía el texto del atributo "mensajes" del objeto compartido a todos los sockets conectados que no hayan cerrado su conexión con el servidor, para ello se usa el array de sockets , de esta manera todos ven la conversación. *Será necesario abrir un stream de escritura a cada socket y escribir el texto.*

Desde el hilo servidor se debe de mostrar en consola los clientes que actualmente hay conectados. Ejemplo para 4 clientes conectados.

Servidor iniciado . . . .
Número de conexiones actuales: 1
Número de conexiones actuales: 2
Número de conexiones actuales: 3
Número de conexiones actuales: 4

public class HiloServidorChat extends Thread {

Atributos

```
DataInputStream fentrada;
Socket socket = null;
ComunHilos  comun;
```

Metodo constructor

public HiloServidorChat(Socket s, ComunHilos comun) {

Otros métodos

public void run()

private void EnviarMensajesaTodos(String texto)

Clase ComunHilos

Esta clase estará compartida por todos los hilos

Atributos de la clase:

- int CONEXIONES : (n.º de conexiones totales, ocupadas en el array)
- int ACTUALES: (n.º de conexiones actuales)
- int MAXIMO: (Máximo conexiones permitidas)
- Socket tabla[] = new Socket[MAXIMO] (Sockets conectados)
- String mensajes (Mensajes del chat)

Constructores

public ComunHilos (int maximo, int acutales, int conexiones, Socket[] tabla)

public ComunHilos()

Métodos modificadores

public int getMAXIMO() *(devuelve el n.º máximo de conexiones permitidas)*

public void setMAXIMO (int maximo) *(Modifica el n.º de conexiones permitidas)*

public int getCONEXIONES()

public synchronized void setCONEXIONES (int conexiones)

public String getMensajes()

public synchronized void setMensajes (String mensajes)

public int getACTUALES()

public synchronized void setACTUALES(int actuales)

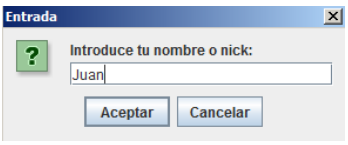
public synchronized void addTabla(Socket s, int i) *(añade socket al array de socket)*

public SocketgetElementoTabla (int i)

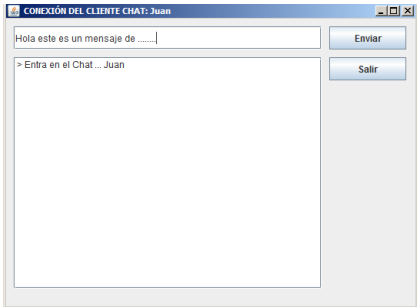
PROGRAMA CLIENTES

Desde el programa cliente se realizan las siguientes funciones:

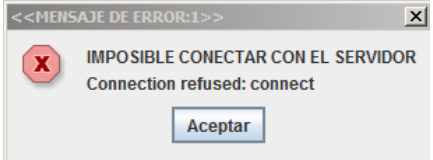
- En primer lugar se pide el nombre que el usuario utilizará para el chat. (Esto se realizará mediante un Jframe que contenga la leyenda "Introduce tu nombre o nick" y un cuadro de texto donde podamos introducir un texto. Además debe de disponer de dos botones uno para aceptar y otro para cancelar.



- Se crea un socket al servidor de chat en el puerto pactado. Si todo es correcto, el servidor asignará un hilo al cliente y se mostrará en la pantalla de chat del cliente la conversación que hay hasta el momento.



- Si existe problema en la conexión, deberá lanzar un mensaje parecido al siguiente:



- El cliente, podrá escribir sus mensajes y pulsar el botón Enviar. Automáticamente su mensaje será enviado a todos los clientes del chat.

- Con el botón “Salir” se finaliza la conexión del cliente al chat , para ello , el cliente enviara un “ * ” al servidor para que este sepa que va ha finalizar la conexión.

Clase ClienteChat

Definimos variables , campos de la pantalla y streams de entrada y salida. La clase debe de extender a JFrame , e implementar ActionListener para controlar la acción de los botones y Runnable para añadir la funcionalidad de hilo a la pantalla, será necesario añadir el método run() con el proceso a realizar por el clientes

Constructores

En el constructor vamos a preparar la pantalla. Se escribe el socket creado y el nombre del cliente de chat; se crean los flujos de entrada y salida “fentrada” “fsalida”. A continuación se escribe en el flujo de salida un mensaje indicando que el usuario ha entrado en el chat . Este mensaje será recibido por el hilo (HiloServidorChat) y lo debe de enviar a todos los clientes conectados.

public ClienteChat (Socket s, String nombre)

Otros métodos

public void actionPerformed (ActionEvent evento) *(Método para controlar la acción de los botones)*

Cuando se pulsa el botón “Enviar” se envía al flujo de salida el mensaje que el cliente ha escrito , si no se escribe nada en el mensaje, éste no se envía; es decir cuando se pulsa el botón enviar si el flujo de salida contiene una cadena vacía este flujo no se envía.

Si se pulsa el botón “Salir” se envía primero un mensaje indicando que el usuario abandona el chat y a continuación un “ * “ indicando que el usuario va a salir del chat.

Método run()

Dentro de este método el cliente lee lo que el hilo servidor le manda (los mensajes de chat) para mostrarlos por el área de texto. Esto se realiza mediante un proceso repetitivo que termina cuando el usuario pulsa el botón “Salir” , que cambiará el valor de la variable repetir a “false” para que finalice el bucle.

Método main ()

En el método main() se realizará lo siguiente:

- Pedir el nombre del usuario.
- Realizar la conexión al servidor.

- Crear un objeto de la clase “ClienteChat”
- Mostrar la pantalla
- Lanzar el hilo cliente.

OBSERVACIÓN PARA LA PRUEBA DEL PROGRAMA

1. Para ejecutar el servidor de chat se necesita que las clases java (ServidorChat, HiloServidorChat y ComunHilos) se encuentren en la misma carpeta. El programa cliente (ClienteChat) puede estar en otra carpeta cualquiera.
2. Debemos de lanzar en primer lugar el Servidor y a continuación ejecutar los clientes.
3. Tanto el programa cliente como el servidor se pueden ejecutar en la misma máquina , pero para su comprobación ejecutaremos la aplicación en tres máquinas distintas (dos clientes y un servidor). En este supuesto, será necesario especificar en el programa cliente, en la creación del socket , la dirección IP del servidor de chat , así como el puerto correspondiente.
   Ejemplo si el servidor tiene la IP 192.168.8.27
   socket = new Socket (“192.168.8.27” , puerto)

SOLUCIÓN

Clase ClienteChat

```
import java.awt.event.*;
import java.io.*;
import java.net.*;
import javax.swing.*;

public class ClienteChat extends JFrame implements ActionListener, Runnable {
	private static final long serialVersionUID = 1L;
	Socket socket = null;
	// streams
	DataInputStream fentrada;
	DataOutputStream fsalida;
	String nombre;
	static JTextField mensaje = new JTextField();

	private JScrollPane scrollpane1;
	static JTextArea textarea1;
	JButton botonEnviar = new JButton("Enviar");
	JButton botonSalir = new JButton("Salir");
	boolean repetir = true;

	// constructor
	public ClienteChat(Socket s, String nombre) {
		super(" CONEXIÓN DEL CLIENTE CHAT: " + nombre);
		setLayout(null);

		mensaje.setBounds(10, 10, 400, 30);
		add(mensaje);

		textarea1 = new JTextArea();
		scrollpane1 = new JScrollPane(textarea1);
		scrollpane1.setBounds(10, 50, 400, 300);
		add(scrollpane1);

		botonEnviar.setBounds(420, 10, 100, 30);
		add(botonEnviar);
		botonSalir.setBounds(420, 50, 100, 30);
		add(botonSalir);

		textarea1.setEditable(false);
		botonEnviar.addActionListener(this);
		botonSalir.addActionListener(this);
		setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);

		socket = s;
```

```
            this.nombre = nombre;
            try {
                  fentrada = new DataInputStream(socket.getInputStream());
                  fsalida = new DataOutputStream(socket.getOutputStream());
                  String texto = " > Entra en el Chat ... " + nombre;
                  fsalida.writeUTF(texto);
            } catch (IOException e) {
                  System.out.println("ERROR DE E/S");
                  e.printStackTrace();
                  System.exit(0);
            }
      }// fin constructor

      // accion cuando pulsamos botones
      public void actionPerformed(ActionEvent e) {
            if (e.getSource() == botonEnviar) { // SE PULSA EL ENVIAR

                  if (mensaje.getText().trim().length() == 0)
                        return;
                  String texto = nombre + "> " + mensaje.getText();

                  try {
                        mensaje.setText("");
                        fsalida.writeUTF(texto);
                  } catch (IOException e1) {
                        e1.printStackTrace();
                  }
            }
            if (e.getSource() == botonSalir) { // SE PULSA BOTON SALIR
                  String texto = " > Abandona el Chat ... " + nombre;
                  try {
                        fsalida.writeUTF(texto);
                        fsalida.writeUTF("*");
                        repetir = false;
                  } catch (IOException e1) {
                        e1.printStackTrace();
                  }
            }
      }// accion botones

      public void run() {
            String texto = "";
            while (repetir) {
                  try {
                        texto = fentrada.readUTF();
                        textarea1.setText(texto);

                  } catch (IOException e) {
```

```
                        // este error sale cuando el servidor se cierra
                        JOptionPane.showMessageDialog(null, "IMPOSIBLE CONECTAR
CON EL SERVIDOR\n" + e.getMessage(),
                                "<<MENSAJE DE ERROR:2>>",
JOptionPane.ERROR_MESSAGE);
                        repetir = false;
                    }
                } // while

                try {
                    socket.close();
                    System.exit(0);
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }// run

            public static void main(String args[]) {
                int puerto = 44444;
                Socket s = null;

                String nombre = JOptionPane.showInputDialog("Introduce tu nombre o nick:");

                if (nombre.trim().length() == 0) {
                    System.out.println("El nombre está vacío....");
                    return;
                }

                try {
                    s = new Socket("localhost", puerto);
                    ClienteChat cliente = new ClienteChat(s, nombre);
                    cliente.setBounds(0, 0, 540, 400);
                    cliente.setVisible(true);
                    new Thread(cliente).start();

                } catch (IOException e) {
                    JOptionPane.showMessageDialog(null, "IMPOSIBLE CONECTAR CON EL
SERVIDOR\n" + e.getMessage(),
                            "<<MENSAJE DE ERROR:1>>",
JOptionPane.ERROR_MESSAGE);
                }

            }// main
}// ..ClienteChat
```

Clase ComunHilos

```
import java.net.Socket;

public class ComunHilos {
        int CONEXIONES; //Nº DE CONEXIONES TOTALES, OCUPADAS EN EL ARRAY
        int ACTUALES;   //NÚMERO DE CONEXIONES ACTUALES
        int MAXIMO;     //MÁXIMO DE CONEXIONES PERMITIDAS
        Socket tabla[] = new Socket[MAXIMO];// SOCKETS CONECTADOS
        String mensajes; //MENSAJES DEL CHAT

       public ComunHilos(int maximo, int actuales, int conexiones,
                                        Socket[] tabla) {
              MAXIMO = maximo;
              ACTUALES = actuales;
              CONEXIONES = conexiones;
              this.tabla = tabla;
              mensajes="";
       }

       public ComunHilos() { super(); }

     public int getMAXIMO() { return MAXIMO;  }
       public void setMAXIMO(int maximo) { MAXIMO = maximo;}


       public int getCONEXIONES() { return CONEXIONES;      }
       public synchronized void  setCONEXIONES(int conexiones) {
              CONEXIONES = conexiones;
       }

       public String getMensajes() { return mensajes; }
       public synchronized void setMensajes(String mensajes) {
              this.mensajes = mensajes;
       }

       public int getACTUALES() { return ACTUALES; }
       public synchronized void setACTUALES(int actuales) {
              ACTUALES = actuales;
       }

       public synchronized void addTabla(Socket s, int i) {
              tabla[i] = s;
       }
       public Socket getElementoTabla(int i) { return tabla[i]; }

}//ComunHilos
```

Clase HiloServidorChat

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

```
public class HiloServidorChat extends Thread {
	DataInputStream fentrada;
	Socket socket = null;
	ComunHilos comun;

	public HiloServidorChat(Socket s, ComunHilos comun) {
		this.socket = s;
		this.comun = comun;
		try {
			// CREO FLUJO DE entrada para leer los mensajes
			fentrada = new DataInputStream(socket.getInputStream());
		} catch (IOException e) {
			System.out.println("ERROR DE E/S");
			e.printStackTrace();
		}
	}// ..

	public void run() {
		System.out.println("NUMERO DE CONEXIONES ACTUALES: " +
comun.getACTUALES());

		// NADA MAS CONECTARSE LE ENVIO TODOS LOS MENSAJES
		String texto = comun.getMensajes();
		EnviarMensajesaTodos(texto);

		while (true) {
			String cadena = "";
			try {
				cadena = fentrada.readUTF();
				if (cadena.trim().equals("*")) {// EL CLIENTE SE DESCONECTA
					comun.setACTUALES(comun.getACTUALES() - 1);
					System.out.println("NUMERO DE CONEXIONES
ACTUALES: " + comun.getACTUALES());
					break;
				}
				comun.setMensajes(comun.getMensajes() + cadena + "\n");
				EnviarMensajesaTodos(comun.getMensajes());
			} catch (Exception e) {
				e.printStackTrace();
				break;
			}
		} // fin while

		// se cierra el socket del cliente
		try {
			socket.close();
		} catch (IOException e) {
```

```
                    e.printStackTrace();
                }

        }// run

        // ENVIA LOS MENSAJES DEL CHAT A LOS CLIENTES
        private void EnviarMensajesaTodos(String texto) {
                int i;
                // recorremos tabla de sockets para enviarles los mensajes
                for (i = 0; i < comun.getCONEXIONES(); i++) {
                        Socket s1 = comun.getElementoTabla(i);
                        if (!s1.isClosed()) {
                                try {
                                        DataOutputStream fsalida = new
DataOutputStream(s1.getOutputStream());
                                        fsalida.writeUTF(texto);
                                } catch (IOException e) {
                                        e.printStackTrace();
                                }
                        }
                } // for

        }// EnviarMensajesaTodos

}// ..HiloServidorChat
```

Clase ServidorChat

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


public class ServidorChat  {
        static final int MAXIMO = 5;//MAXIMO DE CONEXIONES PERMITIDAS

        public static void main(String args[]) throws IOException {
                int PUERTO = 44444;

                ServerSocket servidor = new ServerSocket(PUERTO);
                System.out.println("Servidor iniciado...");

                Socket tabla[] = new Socket[MAXIMO];//para controlar las conexiones
                ComunHilos comun = new ComunHilos(MAXIMO, 0, 0, tabla);

                while (comun.getCONEXIONES() < MAXIMO) {
                        Socket socket = new Socket();
                        socket = servidor.accept();// esperando cliente
```

```
                        comun.addTabla(socket, comun.getCONEXIONES());
                        comun.setACTUALES(comun.getACTUALES() + 1);
                        comun.setCONEXIONES(comun.getCONEXIONES() + 1);

                        HiloServidorChat hilo = new HiloServidorChat(socket, comun);
                        hilo.start();
                }
                servidor.close();
        }//main
}//ServidorChat..
```