46015290 – C/Escalante 9 – 46011 – Valencia – Tlf 961.20.60.30 – Fax 961.20.60.31 – http://ieselgrao.edu.gva.es/

EJEMPLO 03 MÚLTIPLES CLIENTES E HILOS. CHAT TCP

- Introducción:
  - Una situación típica de un servidor que atiende a múltiples clientes es un servidor de chat.
  - Vamos a construir uno sencillo que pueda atender a varios clientes a la vez, 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.
- El programa servidor define 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.
- El objeto ComunHilos contiene los siguientes atributos.
  - 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 MAXIMA.
  - 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 un array para tener control de los clientes y así poder enviarles la conversación
  - String mensajes: Contiene los mensajes del chat.
- El programa servidor:
  - define una variable con el máximo número de conexiones permitidas en el ejemplo se definen 5
  - crea el ServerSocket
  - crea un array para llevar el control de los clientes conectados y crea un objeto de tipo ComunHilos donde se inicializan todas las variables comentadas anteriormente.
  - Se hace 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 almacena en el array, se incrementa el número de conexiones y las conexiones actuales y se lanza el hilo para gestionar los mensajes del cliente que se acaba de conectar.
  - 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 visualice en la pantalla. Esto se hace en el método EnviarMensajesaTodos(). Los mensajes que se envían son los que hay en este momento 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

46015290 – C/Escalante 9 – 46011 – Valencia – Tlf 961.20.60.30 – Fax 961.20.60.31 – http://ieselgrao.edu.gva.es/

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 muestra en consola los clientes que actualmente hay conectados, por ejemplo, se muestra esta salida cuando hay 3 clientes conectados:
  - Servidor iniciado…,
  - NÚMERO DE CONEXIONES ACTUALES: 1
  - NÚMERO DE CONEXIONES ACTUALES: 2
  - NÚMERO DE CONEXIONES ACTUALES: 3

- La clase ComunHilos compartida por todos los hilos tiene los atributos comentados anteriormente y métodos para dar valor y obtener el valor de los atributos. Los métodos definidos como synchronized permitirán que dos o más hilos no interfieran en el estado de los atributos.
- Desde el programa cliente se realizan las siguientes funciones:
  - En primer lugar se pide el nombre que el usuario utilizará en el chat.
  - Se crea un socket al servidor de chat en el puerto pactado. Si todo va bien, 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 no se puede establecer conexión, se visualiza un mensaje de error.
  - El cliente puede escribir sus mensajes y pulsar el botón Enviar, automáticamente su mensaje será enviado a todos los clientes del chat.
  - El botón Salir finaliza la conexión del cliente al chat, enviará un * al servidor para que este sepa que va a finalizar la conexión.
  - Código de la clase ClienteChat:
    - Se definen variables, campos de la pantalla y los streams de entrada y salida.
    - La clase extiende JFrame, que nos permite añadir la interfaz gráfica, e implementa 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 cliente.
    - En el constructor ser prepara la pantalla. Se recibe el socket creado y el nombre del cliente de chat y ser crean los flujos de entrada y salida. A continuación se escribe en el flujo de salida un mensaje indicando que el usuario ha entrado en el chat. Este mensaje lo recibe el hilo HiloServidor Chat y se lo manda a todos los clientes conectados.
    - 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.
    - Si se pulsa el botón Salir, se envía primero un mensaje indicando que el usuario abandona el chat y, a continuación, un asterisco indicando que el usuario va a salir del chat.
    - Dentro del método run(), el cliente lee lo que el hilo servidor le manda (los mensajes del chat) para mostrarlo en el textarea. Esto se realiza en 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.
    - En el método main(), se pide el nombre de usuario, se realiza la conexión al servidor, se crea un objeto ClienteChat, se muestra la pantalla y se lanza el hilo cliente.
- Para ejecutar el servidor de chat, se necesita que las clases java ServidorChat, HiloServidorChat y ComunHilos esté en la misma carpeta. El programa cliente ClienteChat puede estar en cualquier otra carpeta. Primero se ejecuta el programa servidor, y después se ejecutan los clientes.
- En el código, el programa cliente y el servidor se ejecutan en la misma máquina. Pero lo normal es que el servidor esté en una máquina y el cliente en otra. En este caso, es necesario especificar en el programa cliente, en la creación del socket, la dirección IP donde está el servidor de chat.