En este apartado modificaremos ligeramente la aplicación de *sockets stream* que desarrollamos en la lección anterior para que una vez que el cliente envía la palabra "FIN" y cierra su comunicación, no finalice el programa *Servidor*, sino que siga escuchando más peticiones de clientes.

servidor: modificacion

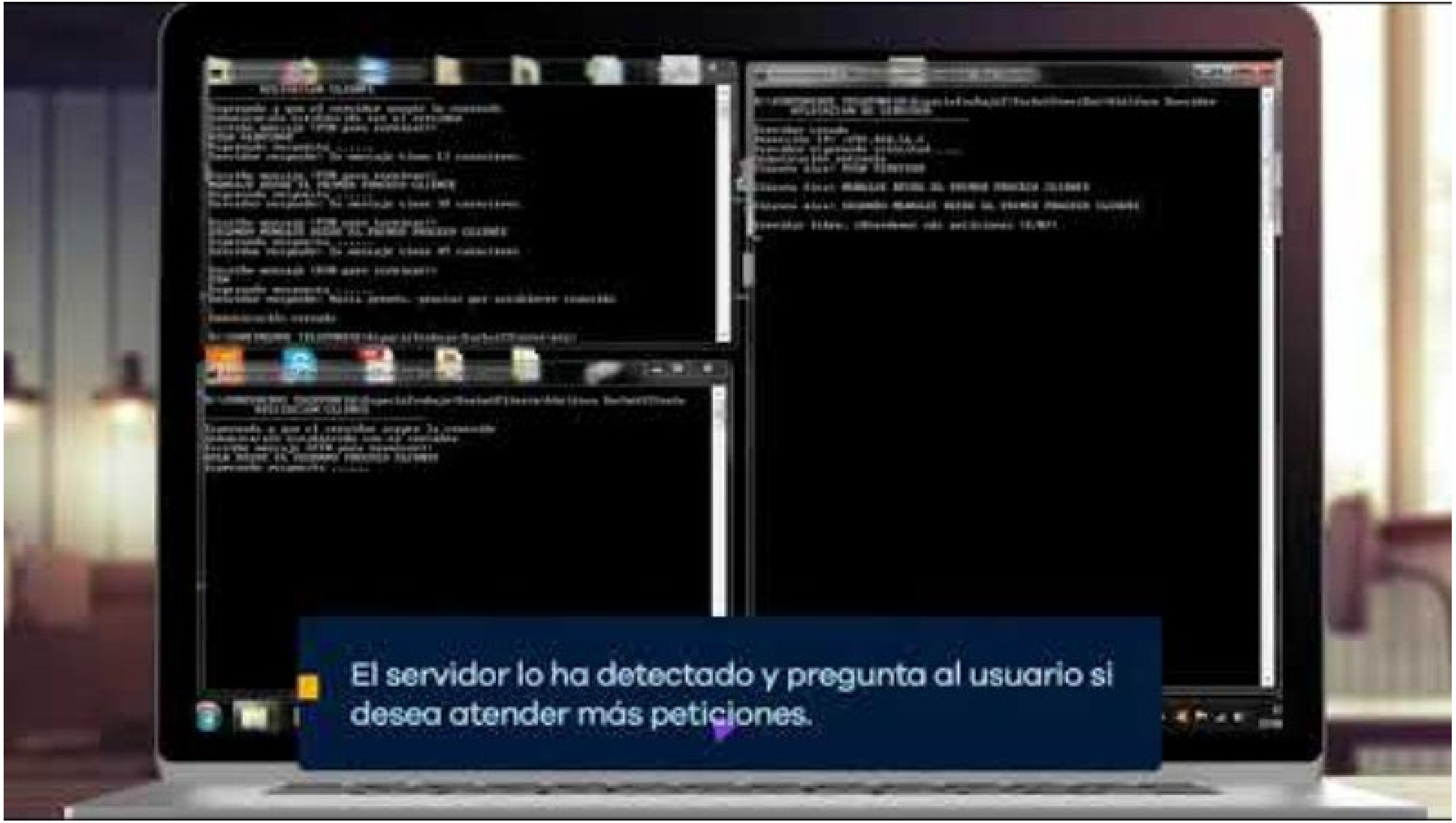
La modificación de la clase *Servidor* será sencilla: una vez que el cliente cierra la comunicación, el usuario podrá decidir si cerra el servidor o continúa escuchando más peticiones a través de una pregunta de continuación.

```
String continuar = "S";
while (continuar.toUpperCase().equals("S"))
        {
        …….
        System.out.println("Servidor libre, ¿Atendemos más peticiones (S/N)?");
        continuar = lector.nextLine();
        }
```

Cliente: modificacion

En los apuntes se indroduce un :

System.out.println("Esperando respuesta ..... ");


El servidor lo ha detectado y pregunta al usuario si desea atender más peticiones.

Has podido comprobar que el servidor no puede atender a varios clientes al mismo tiempo, mientras el servidor tiene establecida una comunicación con un cliente, el segundo cliente queda bloqueado hasta que el primero cierre su comunicación.

Este problema podría solucionarse aplicando multitarea, es decir, varios hilos de ejecución simultáneos, cada uno de ellos atendiendo la solicitud de un cliente.

Servidor multihilo

En este apartado modificaremos la aplicación anterior para lograr que el servidor sea capaz de atender varios *clientes* simultáneos.

Para cada solicitud que recibe un objeto *ServerSocket*, abrirá un objeto *Socket* distinto asociado al cliente que realizó la solicitud. A partir del objeto *Socket* obtenido establecerá el canal de comunicación. Mientras un cliente está siendo atendido, otro cliente puede realizar la solicitud, pero quedará bloqueado a la espera de que el primer cliente cierre la sesión.

Para establecer comunicación simultánea con varios clientes, el *ServerSocket* necesita obtener cada *Socket* cliente en un hilo independiente de ejecución donde también se realizarán todas las operaciones de entrada/salida.

Igual que ocurría antes, el servidor acepta la solicitud de un cliente y recoge el *Socket* solicitante en la referencia *enchufeAlCliente*. Como puedes comprobar, este proceso se repetirá varias veces, ya que lo hemos colocado en un bucle mientras infinito. Para cada solicitud entrante lanzamos un hilo de la clase *HiloEscuchador* al que le pasamos como argumento la referencia al objeto *Socket* solicitante.