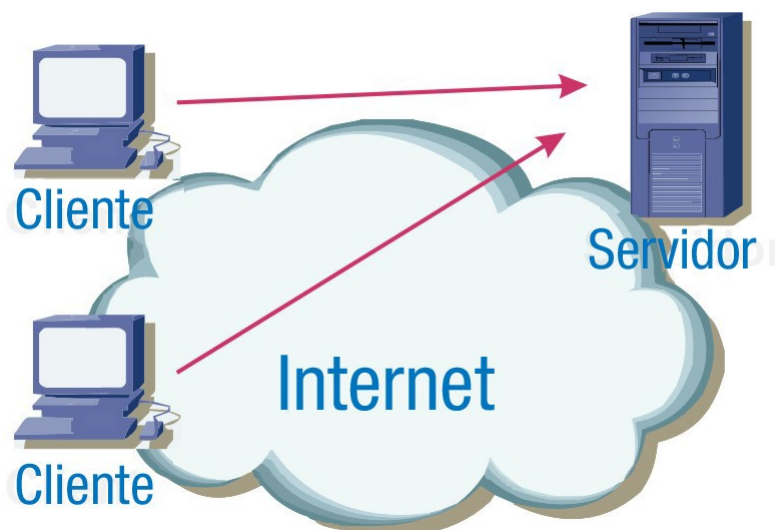


I.E.S Aguadulce

Titulación: Desarrollo de Aplicaciones Multiplataforma

Asignatura: Programación de servicios y procesos



Tarea número 3

– Programación en red –

Autor:	David Jiménez Riscardo
Teléfono:	618882196
E-Mail:	david.jimenez.riscardo@gmail.com

Dos Hermanas, 11 de Enero del año 2022

Table of Contents

1	SERVIDOR.....	III
2	LA CLASE CALCULADORA.....	V
3	CLIENTE.....	VI
4	EXPLICACIÓN DEL FUNCIONAMIENTO.....	VIII
4.1	SERVIDOR CONCURRENT.....	IX
4.2	CLIENTES EN DIFERENTES HILOS.....	XI
5	EVALUACIÓN DE LA TAREA.....	XII

1 Servidor

Implementar un Servidor con sockets en Java con las siguientes características:

- el puerto de escucha TCP se recibirá a través de un parámetro que se le pasa al programa. Si no se le pasa ningún parámetro o bien no es válido, se utilizará 6000 como valor por omisión para el puerto de escucha.
- se aceptarán hasta tres conexiones con clientes simultáneamente.
- el servidor recibirá, de cada cliente, la petición para que le realice una única operación matemática, en el siguiente formato "num1;num2;operación, donde:
 - num1 representa el primer operador (valor real).
 - num2 representa el segundo operador (valor real).
 - operación representa un carácter ('+', '-', '*', '/') con la operación que el cliente quiere que se realice con los números anteriores.
- para el resto de posibles cadenas de caracteres, no ajustadas a ese patrón, enviadas por el cliente, el servidor deberá contestar con un texto de error;

Cuando el servidor inicie su ejecución debe mostrar la siguiente información en su consola:

SERVIDOR CONCURRENTE

SERVIDOR CONCURRENTE

Servidor de XXX YYY

Servidor iniciado.

Escuchando por el puerto ZZZZ.

Esperando conexión con cliente.

donde XXX e YYY serían vuestro nombre y apellidos y ZZZ el puerto por el que ha sido lanzado el servidor. Por ejemplo:

SERVIDOR SECUENCIAL

SERVIDOR CONCURRENTE

Servidor de Francisco Javier MartínezServidor iniciado.Escuchando por el puerto 5400

...

Cuando un cliente se conecte al servidor:

1. En la consola del servidor debe aparecer la siguiente información:

Conexión establecida con cliente.Iniciando hilo servidor X.Recibiendo del CLIENTE:
TTTT

donde X será el número de hilo creado por el Servidor y TTTT sería la operación que el cliente le pide realizar al servidor. Por ejemplo:

```
Conexión establecida con cliente.Iniciando hilo servidor 1.Recibiendo del CLIENTE:  
2;5;+
```

2. El servidor debe enviar al cliente el resultado de la operación que este le ha pedido, como una única cadena de caracteres. Por ejemplo:

```
2.0 + 5.0 = 7.
```

3. Una vez resuelta dicha consulta, el servidor finalizará la ejecución de dicho hilo, mostrando:

```
Hilo servidor X: Fin de la conexión con el cliente
```

donde X será el número de hilo creado por el Servidor.

4. Atendidas las peticiones máximas de clientes, por parte del servidor, este debe de cerrar su ejecución, mostrando el mensaje:

```
Fin de ejecución del servidor.
```

Esto implica la necesidad de que el servidor espere a que todos sus hilos finalicen para sincronizar el fin del mismo.

Una vez realizado ese análisis detallado del comportamiento del servidor, ahora podemos observar un ejemplo completo de los mensajes mostrados por el servidor durante una posible ejecución.

Supongamos que durante la ejecución de un servidor un cliente le solicitara realizar la operación "78.25 * 25.4":

1. Se inicia el servidor configurado en el puerto indicado o en el de por defecto.
2. Se le conecta un cliente.
3. El cliente le envía el mensaje "78.25;25.4;*".
4. El servidor descompone el mensaje para obtener los 3 elementos de la operación (operandos y operador).
5. El servidor calcula el resultado y devuelve al cliente un mensaje con el mismo ("78.25 * 25.4 = 1987.55").

6. El servidor cierra el hilo y queda a la espera de una nueva petición (hasta el número máximo de clientes aceptados).

En tal caso el servidor debería haber ido mostrando la siguiente información por su consola (además de obviamente ir enviando la respuesta apropiada):

```
SERVIDOR CONCURRENTE
-----
Servidor de Francisco Javier Martínez
Servidor iniciado.
Escuchando por el puerto 6000.
Esperando conexión con cliente.
Conexión establecida con cliente.
Iniciado hilo servidor 1.Recibiendo del CLIENTE:
    78.25;23.4;*
Enviada respuesta.
Hilo servidor 1: Fin de la conexión con el cliente.
```

2 La clase Calculadora

Para la implementación de las operaciones que los hilos servidores deben llevar a cabo y enviar a cada cliente, os proporcionamos la clase Calculadora. Para su uso tan solo tendréis que:

- Instanciarla (llamando a su método constructor): Para ello el hilo servidor deberá pasarle por parámetros los elementos de la operación, es decir:
 - num1: double
 - num2: double
 - operacion: char
- Invocar el método getResultado(): Que me devolverá el resultado formateado como se pide en el ejercicio y que el servidor enviará al cliente.

Podemos probar la interacción de esta clase Calculadora con una clase de prueba. Una posible solución podría ser:

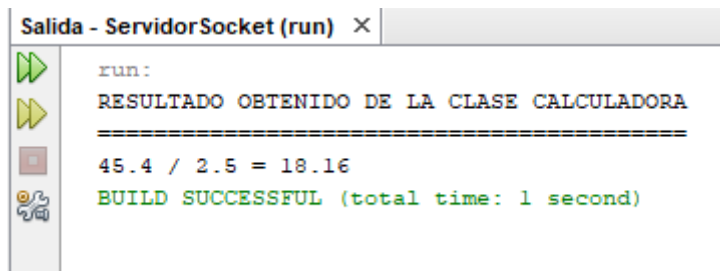
```

package servidoresocket;

/**
 *
 * @author Usuario
 */
public class PruebaCalculadora {
    public static void main(String[] args){
        Calculadora c = new Calculadora(45.4,2.5,'/');
        String resultado = c.getResultado();
        System.out.println("RESULTADO OBTENIDO DE LA CLASE CALCULADORA");
        System.out.println("=====");
        System.out.println(resultado);
    }
}

```

Y el resultado de tal ejecución sería:



```

run:
RESULTADO OBTENIDO DE LA CLASE CALCULADORA
=====
45.4 / 2.5 = 18.16
BUILD SUCCESSFUL (total time: 1 second)

```

3 Cliente

Implementar un cliente que lea del archivo "operaciones.txt" una serie de operaciones (máximo 3 operaciones), que genere tantos hilos como operaciones ha leído. Posteriormente, en cada uno de los hilosCliente debe establecer conexión con un hiloServidor, a través de un Socket comunicar con él para enviar la petición (tal y como se leyó del archivo) y recibir la respuesta.

Cuando el cliente inicie su ejecución debe mostrar la siguiente información en su consola:

```

HILOS CLIENTES
-----
Clientes de XXX YYYY
Leyendo configuración de hilos (operaciones) del archivo de texto...
Cargada configuración de clientes.
Cantidad de clientes: Z

```

donde XXX e YYY serían vuestro nombre y apellidos y Z el número de operaciones del archivo de texto. Por ejemplo:

```
HILOS CLIENTES
-----
Clientes de Francisco Javier Martínez
Leyendo configuración de hilos (operaciones) del archivo de texto...
Cargada configuración de clientes.
Cantidad de clientes: 3
```

Cuando se carguen las operaciones y estemos en disposición de conectar con el servidor, seguiremos la siguiente secuencia de pasos:

1. En la consola del cliente debe aparecer la siguiente información:

```
Ejecución de clientes concurrentes
-----
PROGRAMA CLIENTE INICIADO ...
```

2. El cliente enviará al servidor su operación, tal y como se ha leído desde el archivo de texto. Por ejemplo:

```
2.0;5.0+
```

3. El servidor tratará dicha cadena, creará el objeto Calculadora y obtendrá el resultado. Dicho resultado, expresado como una cadena de caracteres, será enviado al cliente. Por ejemplo:

```
2.0 + 5.0 = 7.0
```

4. Recibido el mensaje del servidor, el cliente deberá mostrarlo. Para ello escribirá en pantalla (para el ejemplo anterior):

```
Recibiendo del SERVIDOR:
      2.0 + 5.0 = 7.0
```

Finalmente, el hiloCliente finalizará su ejecución.

5. Cuando finalicen en su ejecución todos los hilos de clientes, se mostrará el mensaje:

```
Fin del programa principal.
```

Esto implica la necesidad de que el programa espere a que todos sus hilos de clientes finalicen para sincronizar el fin del mismo.

Aquí tienes un ejemplo de cómo podría quedar la ejecución de los hilos clientes conectándose a los hilos servidores que esté en localhost y escuchando por el puerto 6000:

```
HILOS CLIENTES
-----
Cliente de Francisco Javier Martínez
Leyendo configuración de hilos (operaciones) del archivo de texto...

Cargada configuración de clientes.
Cantidad de clientes: 3

Ejecución de clientes concurrentes
-----
PROGRAMA CLIENTE INICIADO ...
PROGRAMA CLIENTE INICIADO ...
PROGRAMA CLIENTE INICIADO ...
Recibiendo del SERVIDOR:
    2.0 + 5.0 = 7.0
Recibiendo del SERVIDOR:
    4.0 + 2.0 = 2.0
Recibiendo del SERVIDOR:
    7.0 + 9.0 = 56.0
Todos los clientes han finalizado su ejecución.
Fin del programa principal.
```


4 Explicación del funcionamiento

Respecto a las capturas de ejecución, es OBLIGATORIO tener como fondo la plataforma con el perfil del alumno/a;

Las pruebas deben realizarse con los ejemplos que se indican en los enunciados, no con otras que se te ocurran a ti.

Cuando se dice “breve explicación del funcionamiento” se pide en efecto ser breve. Indicar tan solo los cuatro o cinco puntos clave importantes en el diseño del programa. Nada más. No más de cinco o seis líneas (no debería ser necesario).

4.1 Servidor concurrente

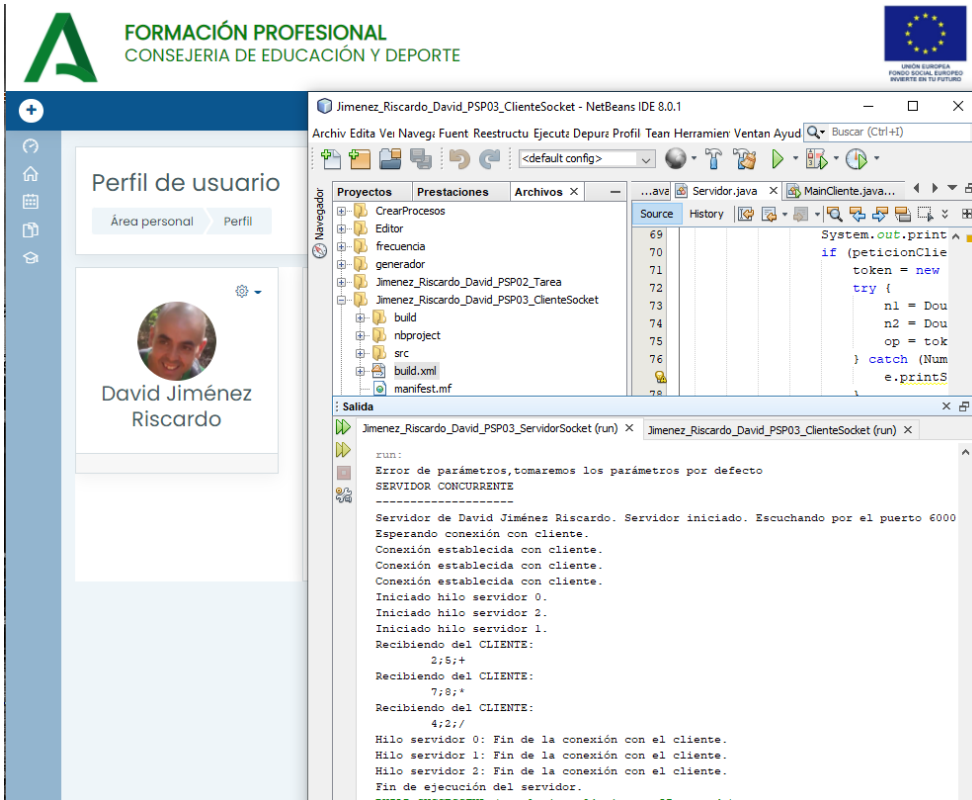
Breve explicación del funcionamiento del programa y capturas del funcionamiento de tu servidor con tu cliente automático utilizando el ejemplo descrito en el enunciado (se inicia el servidor concurrente y se le conectan tres clientes que le van realizando peticiones).

En las capturas deben aparecer varias consolas, la de los clientes y la del servidor, donde se pueda observar cómo ha ido el servidor respondiendo a las peticiones que van realizando cada uno de los tres clientes.

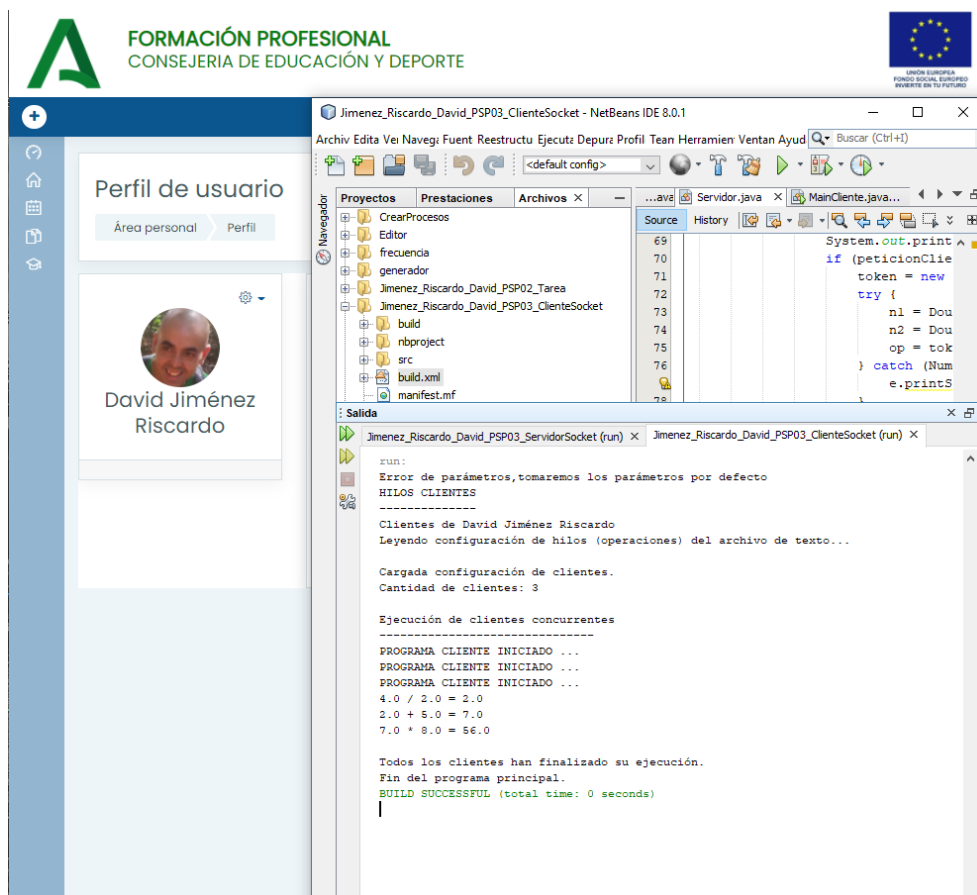
El funcionamiento del servidor puede quedar descrito en los siguientes pasos:

1. Al ejecutar la clase MainServidor estamos instanciando el socket del servidor en un puerto. Dicho puerto podemos tomarlo por línea de comandos o en su defecto sería el puerto 6000.
2. El servidor quedará a la espera de peticiones por parte de los clientes, aceptando un máximo de 3.
3. Por cada petición aceptada nuestro servidor creará e iniciará un hilo quedando almacenados en una estructura de datos.
4. La clase Servidor será la encargada de la instancia de los hilos. Cada hilo tendrá asociado un flujo de entrada y otro de salida.
5. En primer lugar leeremos del flujo de entrada la operación enviada por el cliente, realizaremos el cálculo oportuno a través de la clase Calculadora y escribiremos en el flujo de salida el resultado.
6. Una vez devuelto el resultado de la operación al cliente cerraremos la comunicación con el mismo.
7. Tras finalizar la iteración con todos los clientes cerraremos el socket de servidor.
8. Controlaremos que todos los hilos servidor finalicen antes de finalizar el main.

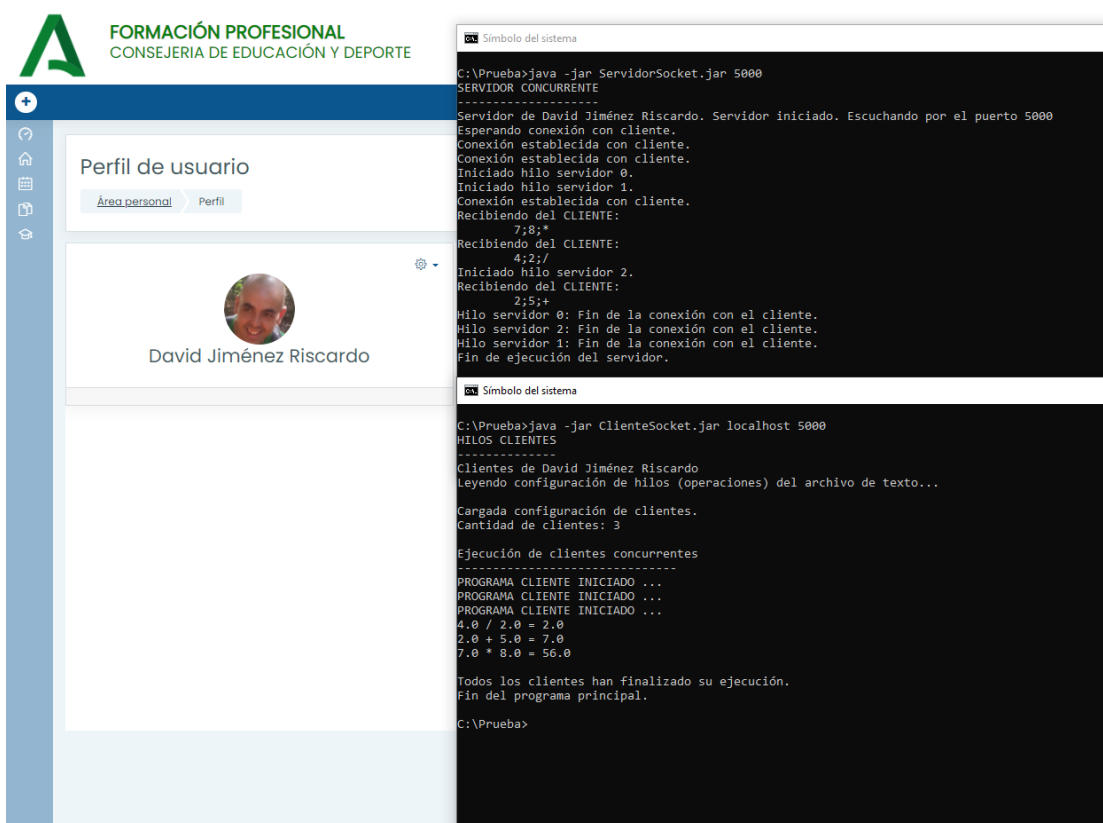
Y podemos ver un ejemplo en las siguientes capturas:



Ejecución de la clase MainServidor desde Netbeans



Ejecución de la clase MainCliente desde Netbeans



Ejecución de ambas clases desde la línea de comandos

4.2 Clientes en diferentes hilos

Breve explicación del funcionamiento del programa que gestiona los clientes que interactúan con tu servidor, utilizando el ejemplo descrito en el enunciado (se inicia el servidor concurrente y se le conectan tres clientes que le van realizando peticiones). Debes explicar tanto el funcionamiento de los clientes, como el programa que se encarga de lanzar los clientes en función de las operaciones que se deben hacer (según el archivo “operaciones.txt”).

El funcionamiento de los clientes puede resumirse en los siguientes pasos:

1. Tras ejecutar la clase MainCliente capturaremos la dirección IP y puerto del servidor por línea de comandos o en su defecto “localhost” y el puerto 6000.
2. Leeremos el contenido del fichero “operaciones.txt” que contiene una operación a realizar por cada línea.
3. Conforme vamos leyendo líneas del fichero, vamos instanciando hilos clientes y almacenándolos en una estructura de datos.
4. La clase Cliente es la encargada de la instancia de los hilos clientes. Cada hilo instanciado contendrá la operación, dirección ip del servidor y puerto.
5. Una vez todos los hilos estén creados, los lanzaremos creando los flujos de entrada y salida.

6. En primer lugar escribiremos la operación a calcular en el flujo de salida y a continuación leeremos del flujo de entrada el resultado.
7. Controlaremos que todos los hilos clientes terminen antes de finalizar el main.

5 Evaluación de la tarea

En esta tabla se muestra la penalización que se aplicará por la no observación de determinados criterios en la elaboración de la tarea, en particular sobre el código Java:

<i>Criterios de evaluación implicados</i>	
a) Se han identificado escenarios que precisan establecer comunicación en red entre varias aplicaciones.	
b) Se han identificado los roles de cliente y de servidor y sus funciones asociadas	
c) Se han reconocido librerías y mecanismos del lenguaje de programación que permiten programar aplicaciones en red.	
d) Se ha analizado el concepto de socket, sus tipos y características.	
e) Se han utilizado sockets para programar una aplicación cliente que se comunique con un servidor.	
f) Se ha desarrollado una aplicación servidor en red y verificado su funcionamiento	
g) Se han desarrollado aplicaciones que utilizan sockets para intercambiar información.	
h) Se han utilizado hilos para implementar los procedimientos de las aplicaciones relativos a la comunicación en red.	
<i>Rúbrica de la tarea</i>	
General: 3.a) Se han identificado escenarios que precisan establecer comunicación en red entre varias aplicaciones.	1 punto/s
General: 3.b) Se han identificado los roles de cliente y de servidor y sus funciones asociadas.	1 punto/s
Cliente: 3.c) y 3.h) Se ha incluido la implementación para la gestión de hilos.	1 punto/s
Cliente: 3.d) Se ha incluido la implementación de sockets.	2 punto/s
Cliente: 3.d), 3.e) y 3.g) Se lleva a cabo la escritura al servidor.	4 punto/s
Cliente: 3.d), 3.e) y 3.g) Se lleva a cabo la escritura del servidor.	4 punto/s
Cliente: 3.d), 3.e) y 3.g) Se cierra los streams y sockets abiertos.	3 punto/s
MainCliente: 3.h) Se crea y utiliza una estructura de datos para almacenar los hilos que se lanzan a ejecución.	3 punto/s
MainCliente: 3.h) Los hilos lanzados se sincronizan en el momento de	3 punto/s

su finalización.	
Servidor: 3.c) y 3.h) Se ha incluido la implementación para la gestión de hilos.	1 punto/s
Servidor: 3.d) Se ha incluido la implementación de sockets.	2 punto/s
Servidor: 3.d), 3.f) y 3.g) Se lleva a cabo la escritura al cliente.	4 punto/s
Servidor: 3.d), 3.f) y 3.g) Se lleva a cabo la escritura del cliente.	4 punto/s
Servidor: 3.d), 3.f) y 3.g) Se cierra los streams y sockets abiertos.	3 punto/s
MainServidor: 3.h) Se crea y utiliza una estructura de datos para almacenar los hilos que se lanzan a ejecución.	3 punto/s
MainServidor: 3.h) Los hilos lanzados se sincronizan en el momento de su finalización.	3 punto/s