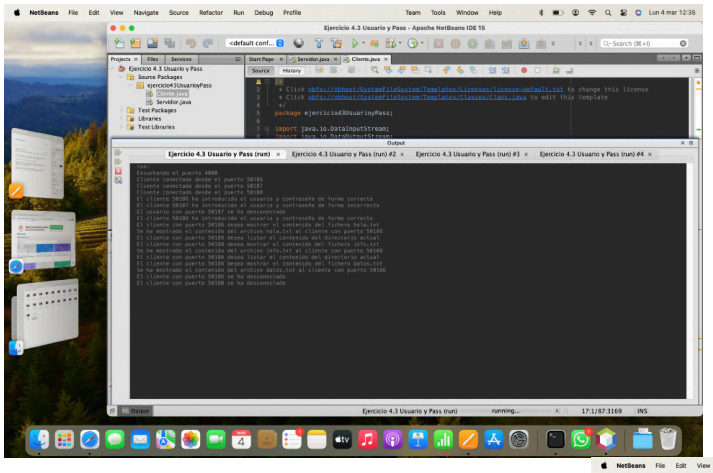
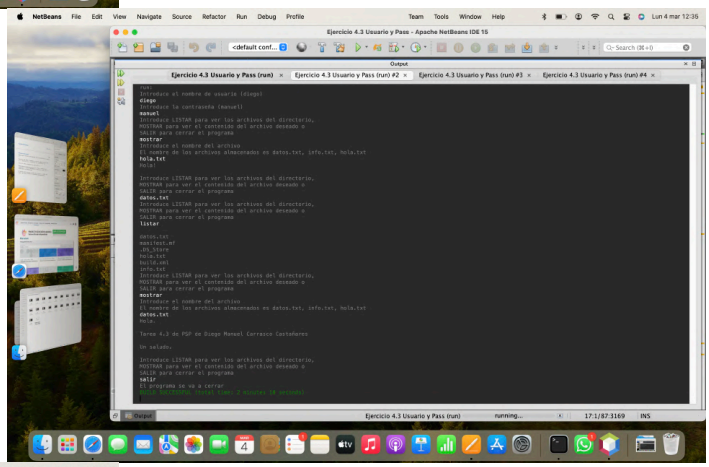
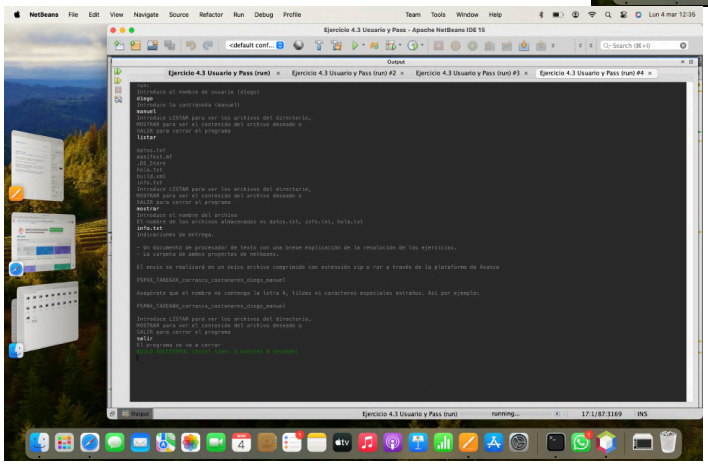
Tarea 4 para Programación de Servicios y Procesos







Diego Manuel Carrasco Castañares

Enunciado.

La tarea de la unidad esta dividida en 3 actividades.

Actividad 4.1. Modifica el ejercicio 1 de la unidad 3 para el servidor permita trabajar de forma concurrente con varios clientes.

Actividad 4.2. Modifica el ejercicio 2 de la unidad 3 para el servidor permita trabajar de forma concurrente con varios clientes.

Actividad 4.3. A partir del ejercicio anterior crea un servidor que una vez iniciada sesión a través de un nombre de usuario y contraseña específico (por ejemplo javier / secreta) el sistema permita Ver el contenido del directorio actual, mostrar el contenido de un determinado archivo y salir.

Para realizar el ejercicio primero debes crear un diagrama de estados que muestre el funcionamiento del servidor.

Criterios de puntuación. Total 10 puntos.

- Actividad 4.1 (2 puntos).
- Actividad 4.2 (2 puntos).
- Actividad 4.3 (6 puntos).

    1. Diagrama de estados - 1 punto.
    2. Cliente – 1 punto.
    3. Servidor – 4 puntos.

Recursos necesarios para realizar la Tarea.

Para realización de la actividad tan sólo es necesario tener instalado el entorno de desarrollo de Java.

Consejos y recomendaciones.

Leer detenidamente el contenido de la unidad.

Indicaciones de entrega.

Debes crear un único fichero comprimido en el que debes incluir el o los proyectos Netbeans y documentos que correspondan a la realización de los ejercicios de la tarea.

Además adjuntarás un documento realizado con un procesador de texto con unas breves explicaciones de cómo has realizado los ejercicios, con las capturas de pantallas que estimes oportunas.

El fichero comprimido debe seguir la nomenclatura PSPXX_TareaZZ_ apellido1_apellido2_nombre, donde XX se corresponde con la unidad y ZZ con el numero de la tarea.

Realiza el envío de dicho archivo comprimido a través de la plataforma. Asegúrate que el nombre no contenga la letra ñ, tildes ni caracteres especiales extraños. Así por ejemplo la alumna Begoña Sánchez Mañas para la primera unidad del MP de PSP, debería nombrar esta tarea como…

sanchez_manas_begona_PSP04_Tarea

INDICE:

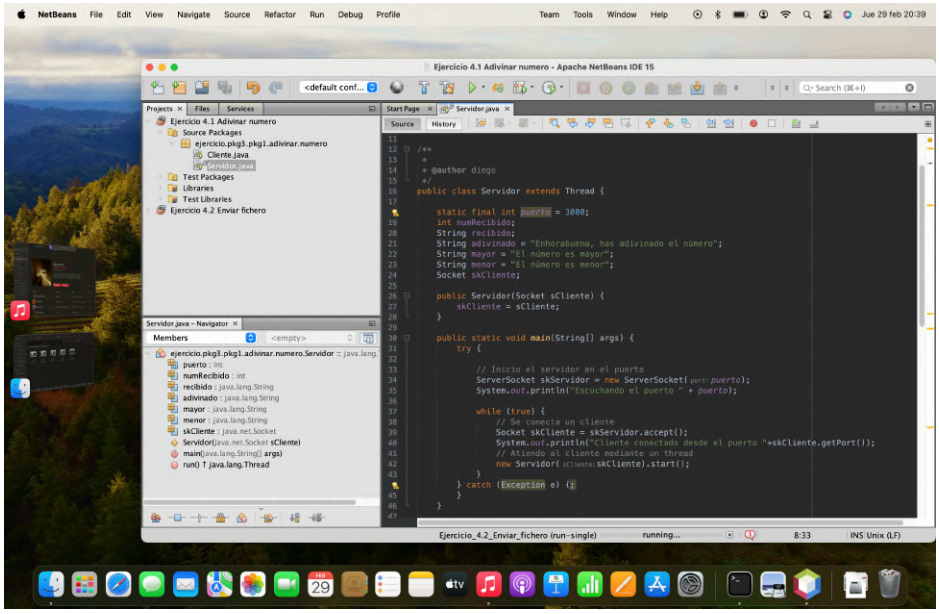
Empezaremos copiando los proyectos de los ejercicios de la tarea 3.

Tras ello los vamos a renombrar, ya que su nombre es *Ejercicio 3.1… y Ejercicio 3.2…,* a *Ejercicio 4.1…* y *Ejercicio 4.2…*

Tras ello vamos a modificar el ejercicio 4.1.

Empezaremos haciendo que extienda de *Thread* y contendrá un constructor al cual le pasaremos el socket del cliente, de un método main y su método run.

El método main iniciará el socket del servidor, a través del puerto que hemos fijado en la variable correspondiente, mostrará el puerto desde el que se ha conectado el cliente, e iniciara dicho servidor.



El método run dispondrá de una variable tipo random, donde instanciará, cada vez que se inicie dicho método, un número random entre el 1 y el 500. Dicha variable de tipo final.

Tras ello, dispondrá del DataImputStream (llamado in) y del DataOutputStream (llamado out) a los cuales le pasaremos por parámetros el ServerSocket del cliente.

A continuación dispondrá de un while, iniciado a verdadero, que contendrá la variable *recibido* que se instanciará con el numero recibido desde la clase cliente a través del socket.

Dispondrá de un *sout* que mostrará el número que ha dicho el cliente y una variable para pasar a int dicho número.

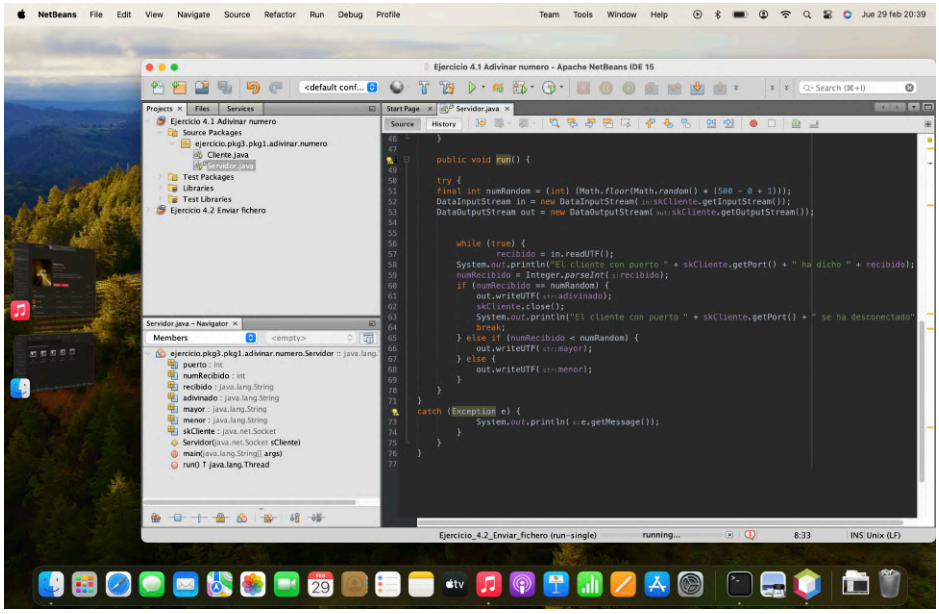
Tras ello dispondrá de un *if*, que comprobará si el número recibido es igual al número random generado, y si es así, lanzará al cliente el string contenido en adivinado (Enhorabuena, has

adivinado el número), cerrará el socket del cliente y mostrara el mensaje que el cliente se ha desconectado.

Si no es así, entrara en un *else-if* en el cual comprobará si el número recibido es menor que el generado y lanzara el string contenido en la variable mayor al cliente (El número es mayor).

Si no corresponde a ninguna de las dos anteriores lanzara el string contenido en la variable menor (El número es menor).

Todo el código contenido en el run estará envuelto en un try - catch para controlar las posibles excepciones.



A continuación, dejaremos tal cual estaba la clase cliente.

En dicha clase voy a crear una variable de tipo string y final llamada host, la cual contendrá la ruta del host, que en mi caso será localhost.

Crearemos otra variable de tipo int y final llamada puerto, que contendrá el número de puerto, cuyo valor será 3000 tal como indica el ejercicio.

Crearemos otra variable de tipo string llamada recibido, que contendrá el mensaje recibido desde la clase servidor.

Crearemos otra variable de tipo string llamada adivinado, que contendrá la cadena “Enhorabuena, has adivinado el número”.

Por último crearemos un scanner para recoger por teclado lo que introduce el usuario llamado sc.

Lo siguiente será crear el método constructor.

Dentro de él crearemos un Socket llamado sCliente, al cual le

pasaremos el host y el puerto.

Crearemos un DataImputStream llamado in que recogerá lo que contenga el socket sCliente mediante el método getInputStream.

También crearemos un DataOutPutStream llamado out que será el encargado de instanciar en el socket sCliente lo que necesitemos mediante el método getOutputStream.

Posteriormente le indicaremos al usuario que introduzca un número del 0 al 500.

Crearemos un while con la condición verdadero (true), y dentro del while le indicaremos al usuario el mensaje “Introduce el número que creas”.

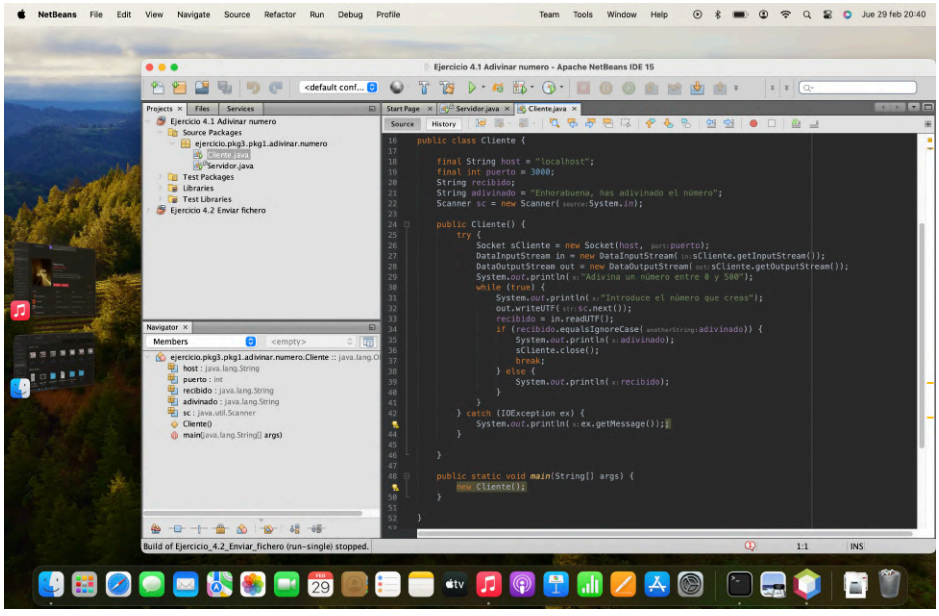
El número introducido lo instanciaremos en la variable out mediante el método writeUTF pasándole la entrada del scanner creada anteriormente.

Tras ello crearemos un if que comprobará, mediante le método equalsIgnoreCase, si la variable recibido es igual a la variable adivinado y si es así mostrará el mensaje almacenado en adivinado al usuario y cerrara el socket sCliente.

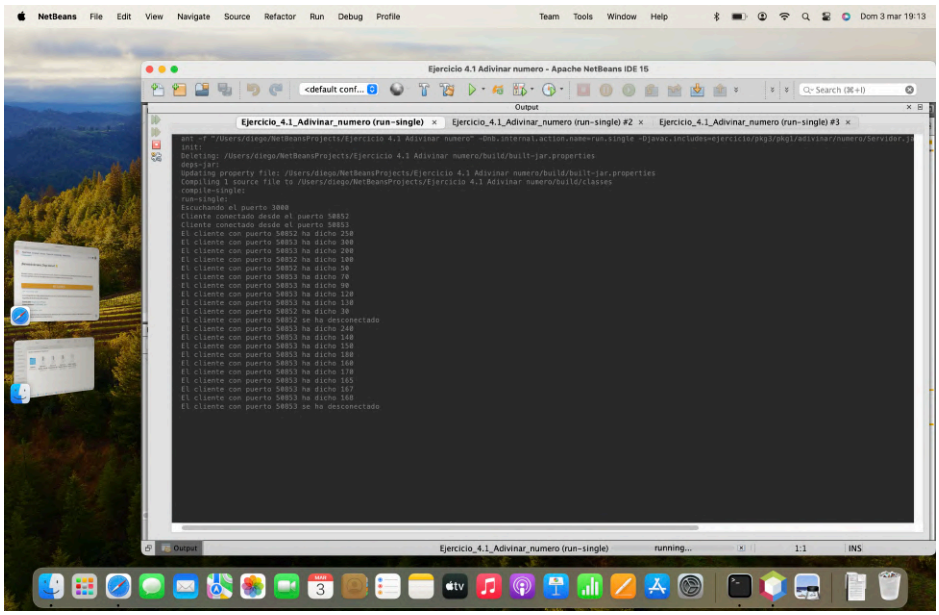
En Caso contrario mostrará por pantalla el contenido de la variable recibido, que contendrá el mensaje recibido desde la clase servidor mediante el socket sCliente.

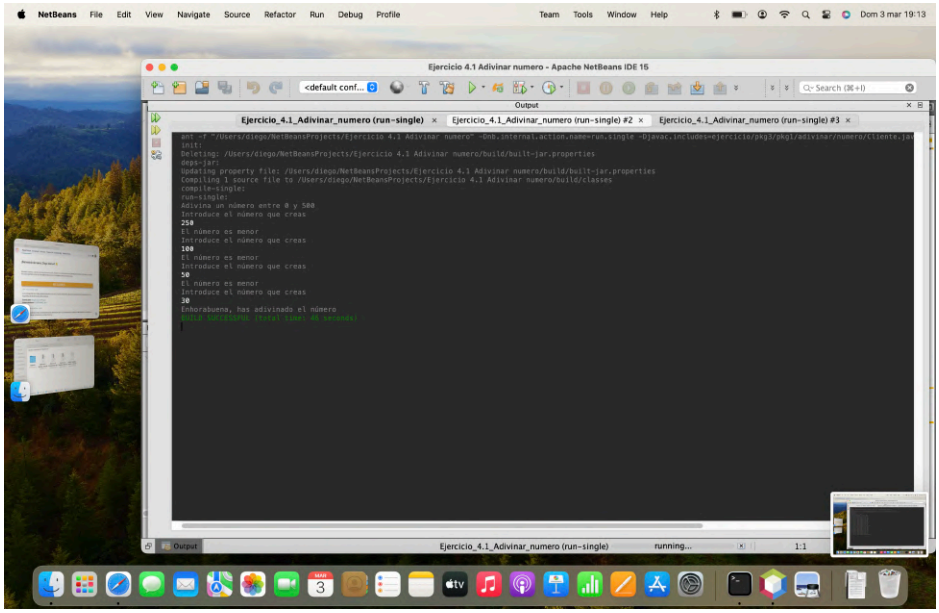
Todo esto estará envuelto en un try – catch.

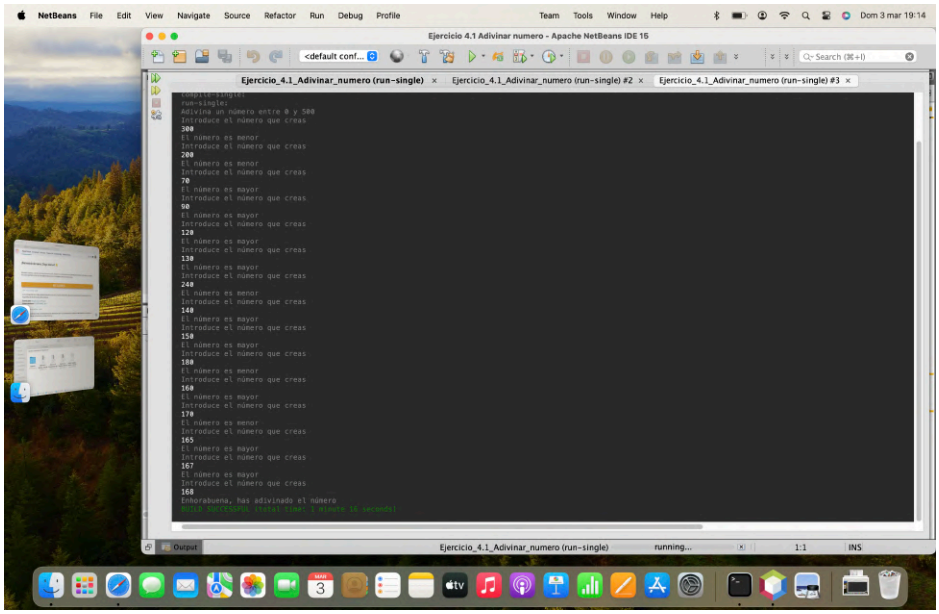
Por último crearemos el main, desde donde llamaremos al método constructor de la clase.



En las siguientes imágenes podemos el output de NetBeans.







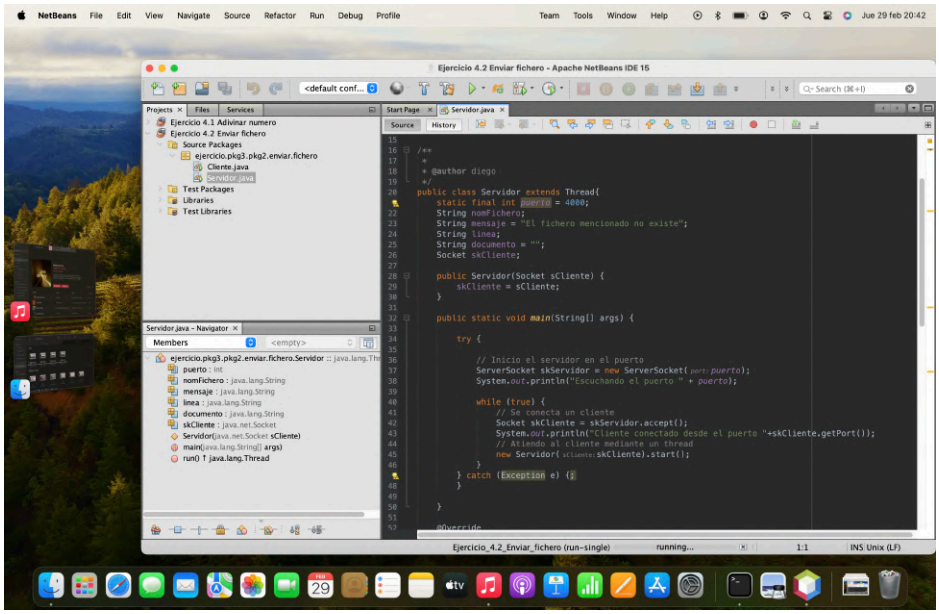
A continuación vamos a componer el ejercicio 4.2

En la clase servidor crearemos las variables correspondientes, un constructor al que le pasaremos el socket del cliente, su método main y el método run.

En el método main crearemos el ServerSocket en el puerto indicado y mostraremos por pantalla el puerto desde el que esta escuchando el servidor.

Tras ello crearemos un while, con la condición verdadero, el cual aceptará el socket del cliente mediante el método accept, mostrara desde que puerto se ha conectado el cliente e iniciara el servidor correspondiente.

Todo ello envuelto en un try - catch para controlar las posibles excepciones.



En el método run crearemos el DataImputStream (llamado in) y el DataOutputStream (llamado out), que se instanciaran con el socket del cliente.

Tras ello recuperaremos el nombre del fichero enviado por el cliente y lo guardaremos en su variable correspondiente.

A continuación mostraremos el nombre del fichero que desea leer el cliente.

Seguidamente crearemos un File llamado fichero, al cual le pasaremos el nombre del fichero dado por el cliente.

Posteriormente crearemos un *if*, que comprobara si el fichero existe, y de ser así, devolverá un booleano al cliente con el valor *true*.

Seguidamente crearemos un FileReader, al cual le pasaremos el fichero creado anteriormente y un BufferedReader al cual le pasaremos el FileReader.

Tras ello crearemos un while, que tendrá como condición que mientras que la siguiente linea a leer no sea *null* vaya sumando el contenido de la variable linea al contenido de la variable documento y así componer el texto completo.

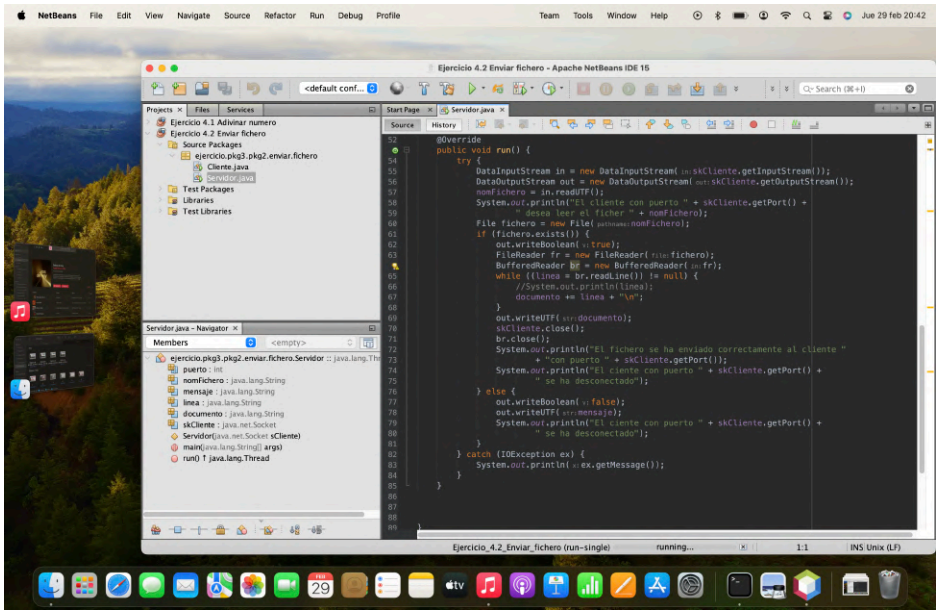
A continuación pasaremos el contenido de la variable documento al cliente, cerraremos el socket del cliente y el BufferedReader.

Posteriormente mostraremos que el archivo se enviado correctamente al usuario y que dicho usuario se ha desconectado.

Si no se cumple la condición del *if* anterior, el boolean que enviaremos al cliente será con valor falso, le enviaremos el string contenido en la variable mensaje (El fichero mencionado no existe) y mostraremos el mensaje de que el cliente se ha desconectado.

Todo ello estará envuelto en un try - catch para controlar las posibles excepciones.

El código de dicha clase podemos verlo en la imagen siguiente.



La clase cliente, en este caso no la hemos modificado, quedado de la siguiente manera.

Crearemos una variable de tipo int y final llamada puerto, que contendrá el número de puerto, 4000 en este caso, tal como indica el ejercicio.

También crearemos una variable de tipo string y final llamada host que contendrá el host, que en nuestro caso será localhost.

Crearemos una variable de tipo boolean llamada existe que nos dirá si existe o no el fichero (a través de la clase servidor).
Tras ello crearemos un Scanner llamado sc.

A continuación crearemos el constructor.

Dentro de él crearemos un socket llamado sCliente al cual le pasaremos el host y el número de puerto (creados anteriormente).

Lo siguiente será crear el DataInputStream y el DataOutputStream a los que llamaremos int y out respectivamente.

Serán instanciados con sus métodos correspondientes (getInputStream y getOutputStream) aplicados al socket sCliente.

Luego mostraremos un mensaje al usuario para que introduzca el nombre del fichero por teclado, siendo el mensaje en cuestión “Introduce el nombre del fichero”.

A continuación crearemos un while con la condición true.

Dentro de dicho while instanciaremos en out, mediante el método write.UTF lo que introducido el usuario por teclado.

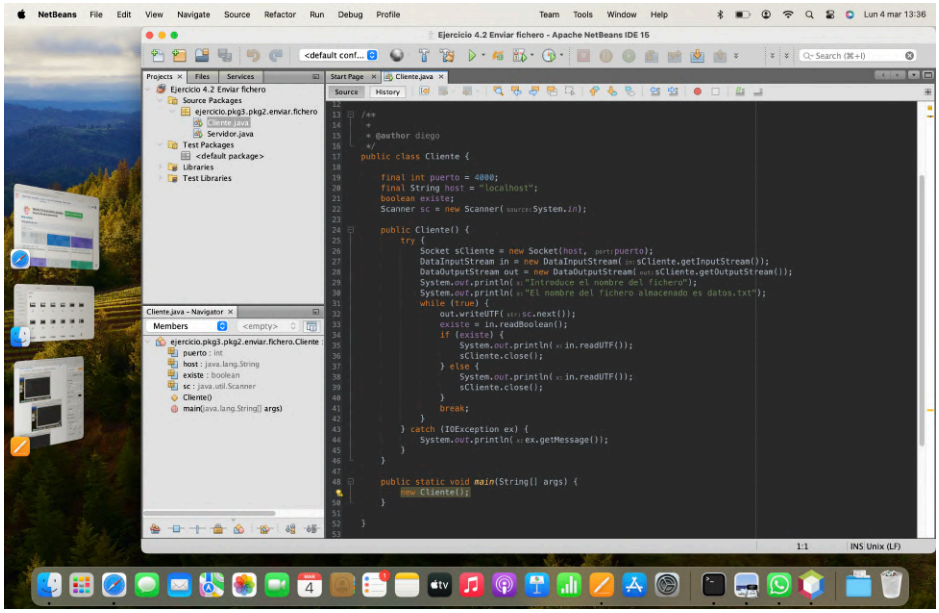
La variable existe la instanciaremos mediante el método readBoleean aplicado a la variable in.

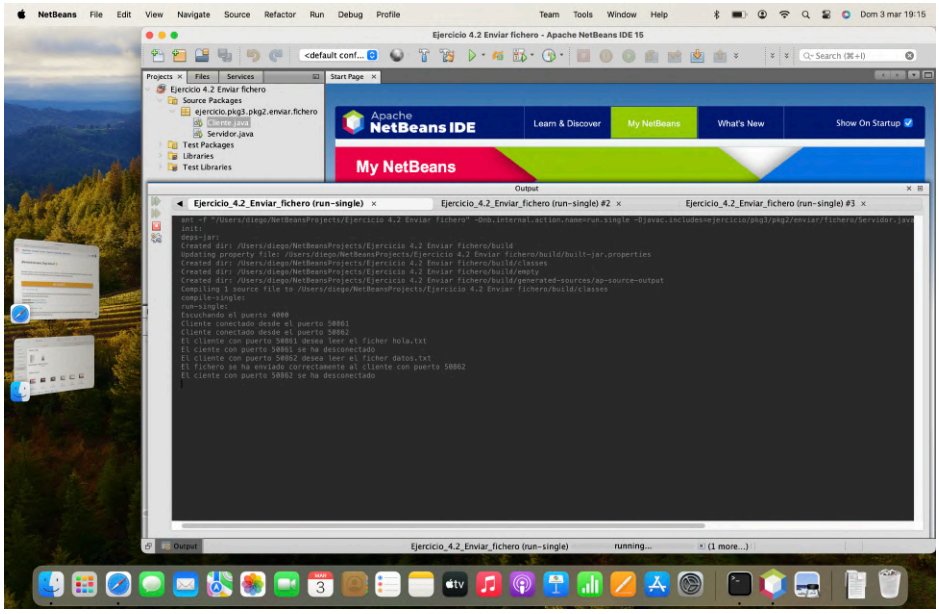
Lo siguiente será crear un if, cuya condición será la variable existe, que si es verdadera, mostraremos por pantalla el contenido de la variable in aplicándole el método readUTF y cerraremos el socket sCliente.

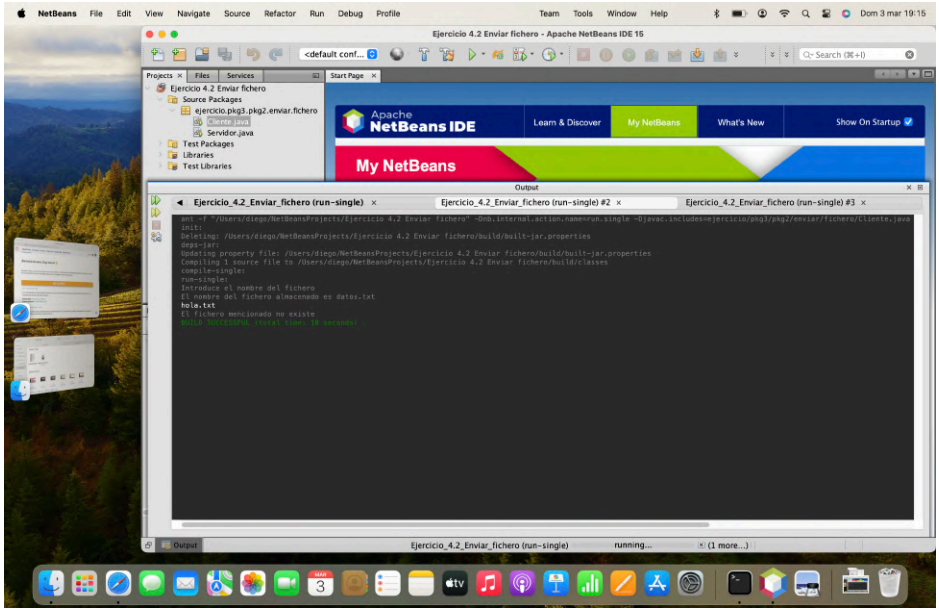
En caso contrario mostrara por pantalla el contenido de la variable in aplicando el método readUTF y cerrara el socket sCliente.

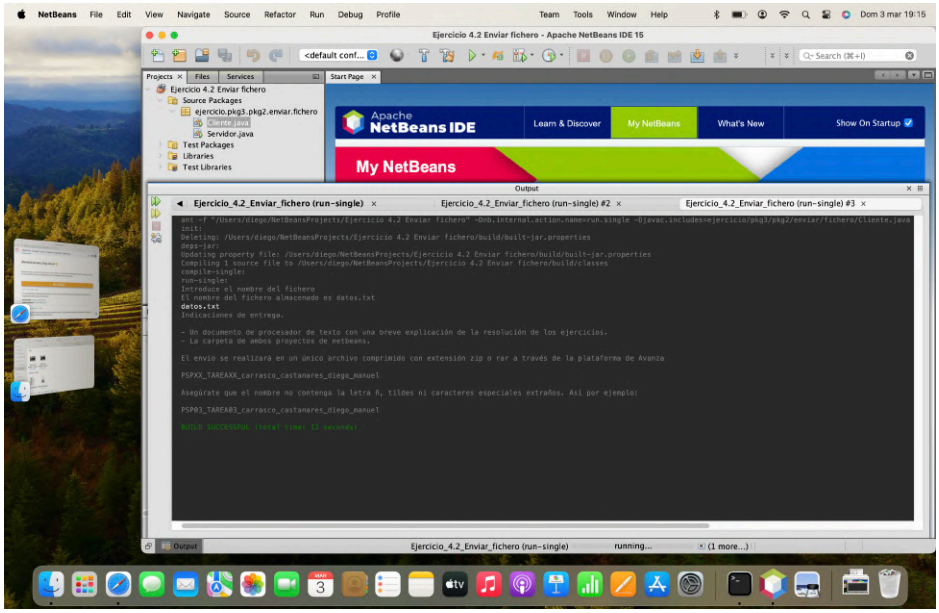
Todo este código estará envuelto en un try – catch.
Por último crearemos el main donde llamaremos al método constructor de la clase.

El código de dicha clase, y el output de NetBeans podemos verlo en las imágenes siguientes.









Ejercicio 4.3. Usuario y pass.

Para la tarea 4.3 empezaremos creando el diagrama de estados, en el cual podemos ver el estado inicial, que es el estado 0.

Tras el pasaríamos al estado 1, en el cual el usuario tiene que introducir un comando. Si el comando no existe volverá a pedir nuevamente que se introduzca un comando.

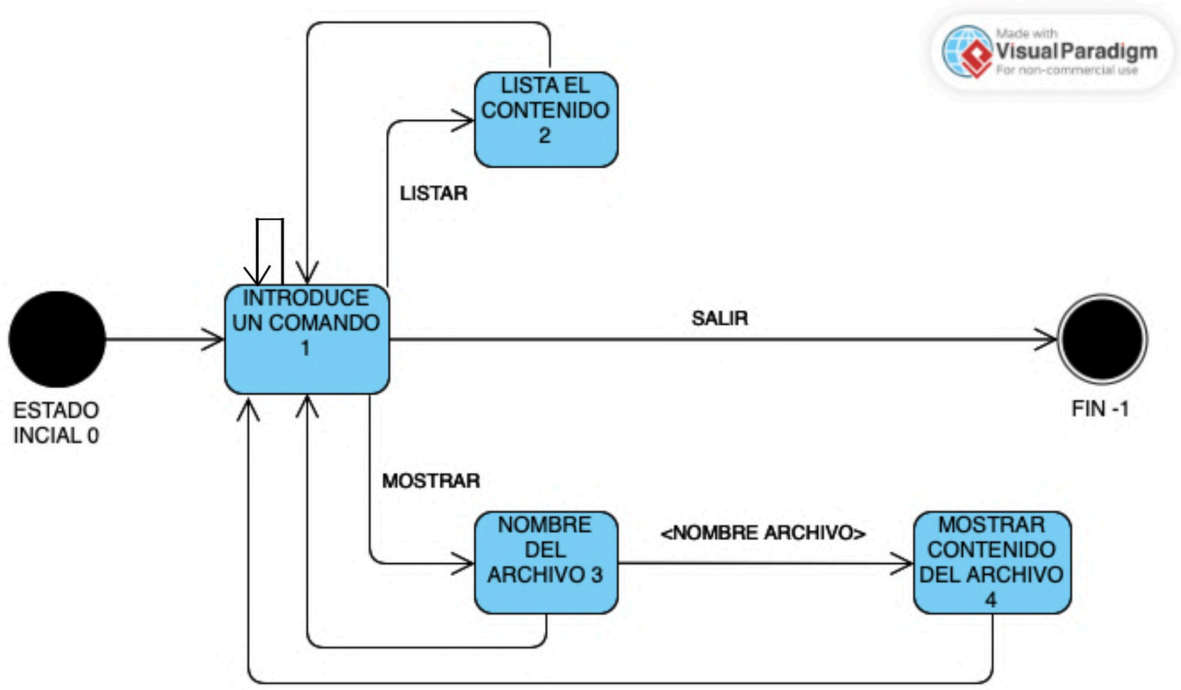
El estado 2 corresponde al comando listar el contenido del directorio actuales, que una vez listado, volveríamos al estado 1.

El estado 3 corresponde al comando mostrar, y pedirá al usuario el nombre del archivo. Si el archivo no existe volveríamos al estado 1, en caso de existir el nombre del archivo pasaríamos al estado 4.

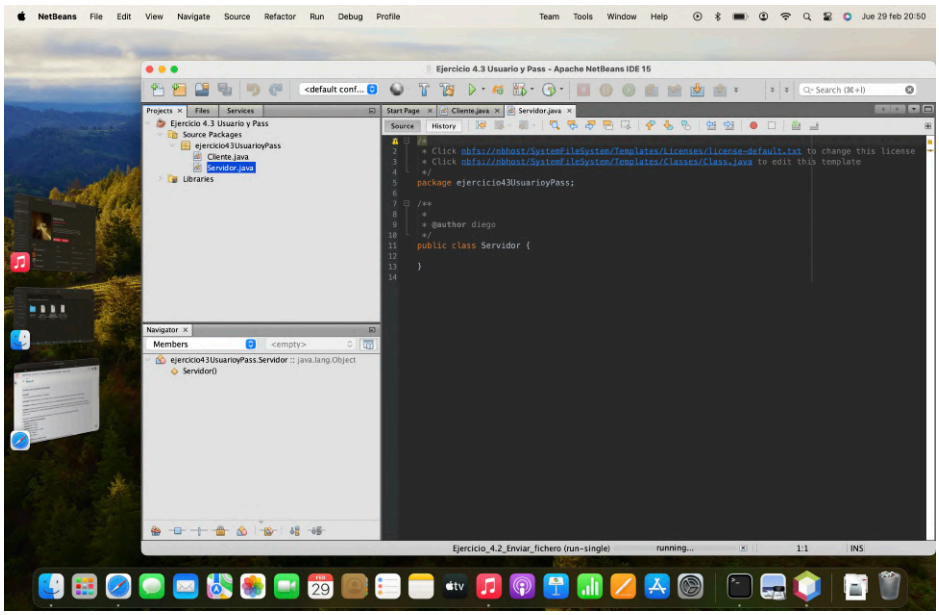
El estado 4 corresponde a mostrar el contenido del archivo que le hemos indicado en el estado 3. Una vez mostrado volveríamos al estado 1.

El estado -1 corresponde al comando salir, y finalizaría la ejecución del programa.

En la siguiente imagen podemos ver el diagrama de estados.



A continuación crearemos las clases servidor y cliente.



La clase servidor extenderá de *Thread*, tendrá varios atributos privados tales como el socket, encargado de almacenar el socket del cliente y user y pass de tipo String, encargados de almacenar el usuario y la contraseña que dará acceso al programa respectivamente.

Tras ello contendrá el método constructor al cual le pasaremos el socket del cliente.

Lo siguiente que haremos será crear el método run, que contendrá don variables de tipo string, una llamada linea, que almacenará cada linea leída del archivo y otra documento, que instalaremos a vacio (usando las comillas, que de no hacerlo nos mostraría la palabra *null* al principio del texto que nos debe de mostrar).

A continuación crearemos el DataOutputStream y el DataImputStream, llamados out e in respectivamente, los cuales instanciaremos a *null.*

Posteriormente, ya dentro del try - catch - finally, instanciaremos los DataImputStream y OutputStream pasándole a ambos, el socket del cliente.

Tras ello lanzaremos el mensaje “Introduce el nombre de usuario (diego)” al cliente, recogeremos el string introducido por el usuario y lo guardaremos en la variable userCliente.

Posteriormente haremos lo mismo que en el apartado anterior pero para el pass, lanzando el mensaje “Introduce la contraseña (manuel)” y guardando el dato en la variable passCliente.

Lo siguiente que haremos será crear un if, que comprobara si el usuario y la contraseña introducidos por el cliente son iguales a los guardaos en el programa (variables user y pass), y si es así, mostraremos que el usuario ha introducido correctamente el usuario y la contraseña.

A continuación devolveremos un boolean con el valor true al cliente e instanciaremos a false el boolean salir.

Tras ello crearemos un while, que tendrá la condición de si salir es false, ejecutará los el código contenido en el.

Dicho while mostrará al usuario un string indicando los comandos disponibles y recogerá el comando introducido por el usuario para guardarlo en la variable comando.

Posteriormente crearemos un switch, al cual le pasaremos la variable comando, convertida a minúscula.

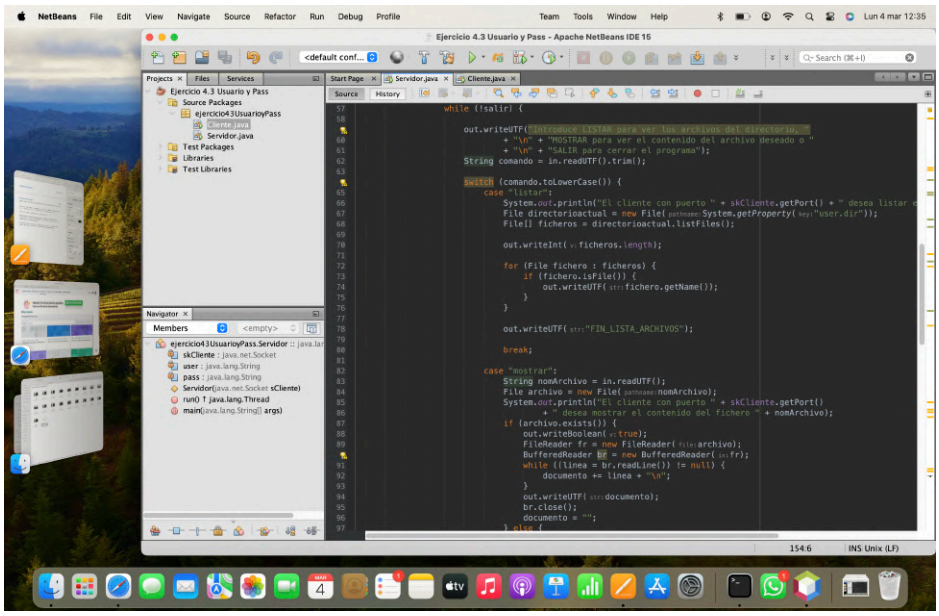
Dicho switch contendrá tres case, uno para el comando listar, otro para el comando mostrar y otro para el comando salir.

El comando listar mostrará un mensaje indicando que el cliente desea listar el contenido. Contendrá un File llamado directorioactual, que extraerá del sistema las dirección del proyecto.

También dispondrá de un array de tipo File, llamado ficheros, que almacenará el contenido de la variable directorioactual.

Tras ello le diremos al cliente la longitud de dicho array.
A continuación crearemos un bucle for que recorrerá el array e irá pasando al cliente el nombre de cada fichero encontrado.

Tras ellos comunicaremos al cliente el mensaje "FIN_LISTA_ARCHIVOS" y haremos el break de este case.



El siguiente case será el de mostrar el contenido de un archivo.

Para ello crearemos un una variable tipo string llamada nomArchivo que se instanciará con la lectura del contenido que nos ha enviado el cliente.

Tras ello dispondrá de un File llamado archivo, al cual le pasaremos el nombre del archivo y mostraremos un mensaje indicando el puerto del cliente y el archivo que quiere leer.

Posteriormente crearemos un if que comprobará si el archivo existe, y si es así, devolverá un boolean al cliente con el valor true, creará un FileReader pasándole el archivo en cuestión y un BufferedReader pasándole el FileReader.

Tras ello crearemos un while, dentro del if anterior, que mientras que la linea leída sea distinta de null, añada a la variable documento el contenido de cada línea con un salto de linea entre ellas.

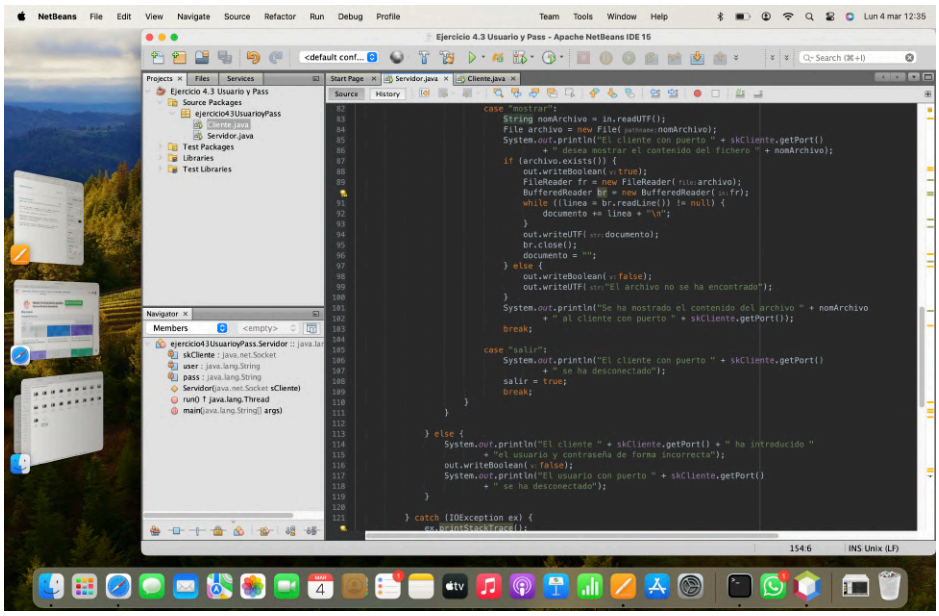
Fuera del while le pasaremos al cliente el documento, cerraremos el BufferedReader e instanciaremos a vacío (usando comillas dobles) la variable documento.

En caso de no existir el archivo el boolean que le pasaremos al cliente será con el valor false y mostraremos un mensaje indicando que el archivo no se ha encontrado.

Tras ello, fuera del else anterior, mostraremos el cliente y el archivo que desea leer y haremos su break.

El último case será salir, que mostrara que el cliente se ha desconectado, cambiara a true el boolean salir y hará su break.

Todo ello estará envuelto en try – catch – finally para capturar las excepciones y cerrar los DataImputStream y OutputStream.



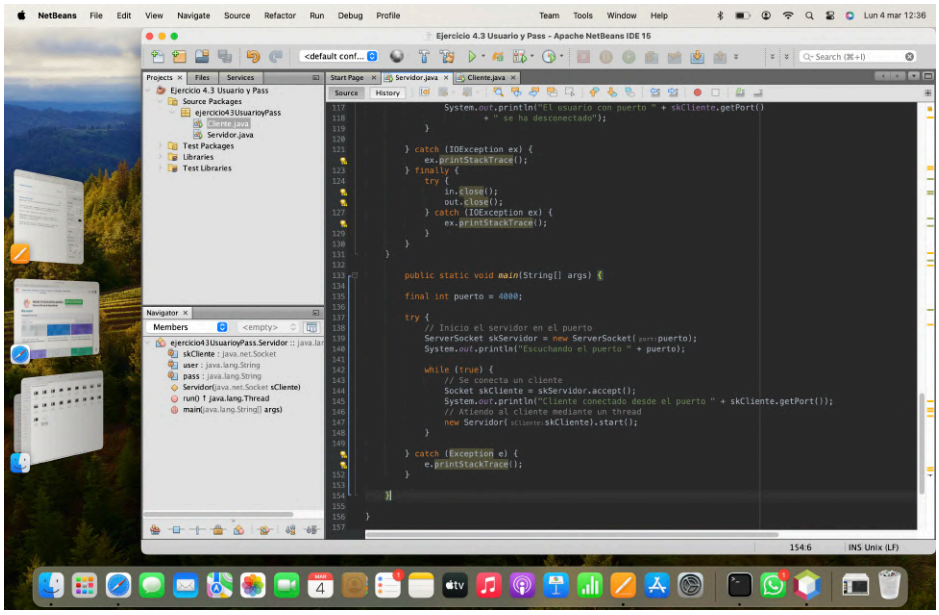
Por último crearemos el main.

Estará compuesto por una variable de tipo final con el número de puerto.

Tras ello crearemos un ServerSocket al cual le pasaremos el número de puerto y mostraremos que estamos escuchando desde el puerto en cuestión.

A continuación crearemos un while, que tendrá como condición true, que aceptará el socket del cliente, mostrará que se ha conectado dicho cliente e iniciará un nuevo servidor con el socket del cliente.

Todo ello envuelto en un try – catch para controlas las excepciones.



Tras ello compondremos la clase cliente.

Estará compuesta por una variable final de tipo int que contendrá el puerto y otra final de tipo string que contendrá el host, en nuestro caso 4000 y “localhost” respectivamente.

Tras ello crearemos el constructor de la clase, el cual creará un socket al cual le pasaremos el host y el puerto.

Después crearemos el DataImputStream, el DataOutputStream y un scanner para leer lo que el cliente introduce por teclado.

Tras ello crearemos una variable llamada mensaje, de tipo string llamada mensaje que se instanciara con el contenido que nos envía el servidor y tras mostraremos el contenido de dicha variable.

A continuación crearemos otra variable de tipo string llamada usuario, que se instanciara con lo leído por teclado y se lo enviaremos al servidor.

Seguiremos los mismos pasos para la contraseña, es decir, mostraremos el mensaje recibido desde el servidor, crearemos una variable de tipo string llamada password que se instanciará con el dato introducido por el cliente y enviara al servidor el contenido de dicha variable.

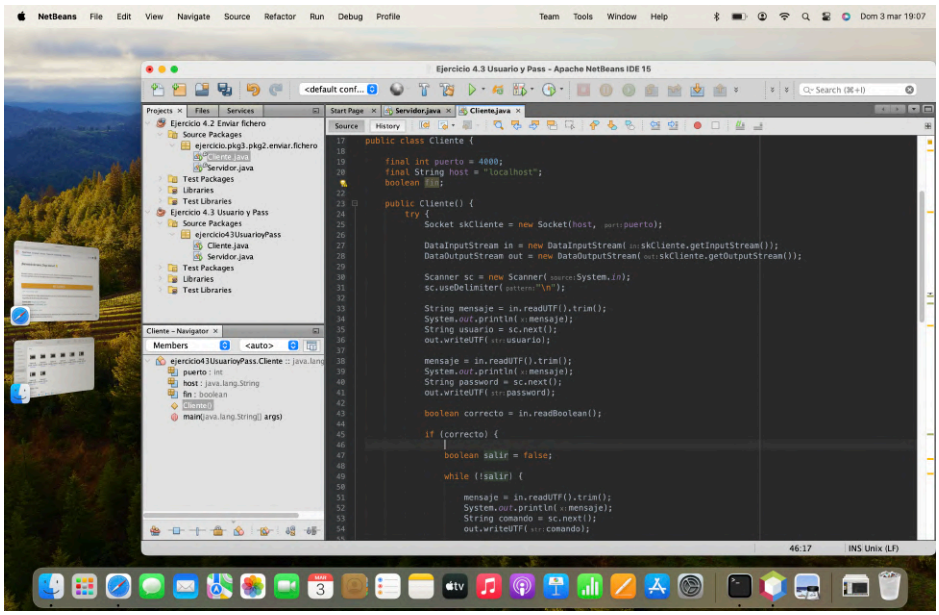
Cabe decir, que posteriormente renombre estas variables a user y pass respectivamente.

Tras ello crearemos una nueva variable de tipo boolean llamado correcto que se instanciará con el contendido recibido por el

servidor (true en el caso de ser correcto el usuario y la pass o false en caso contrario).

Posteriormente crearemos un if que tendrá como condición el boolean anterior, que en el caso de ser true, creara un nuevo boolean llamado salir instanciado a false.

Lo siguiente será crear un while, que mientras el boolean salir sea false, leerá el contenido enviado desde el servidor, lo mostrará, creara una variable de tipo string llamado comando que será instanciada por el dato introducido por el cliente y pasará al servidor el contenido de dicha variable.



A continuación crearemos un switch, que tomará como valor la variable comando y la convertirá a minúsculas para evitar errores de tipo y dispondrá de los tres case que creamos en el servidor (listar, mostrar y salir).

En el case de listar creará un boolean llamado finArchivos y lo instanciará a false.

Dispondrá de un while, que mientras la variable finArchivos sea false, creará una variable de tipo string llamada nombreFichero y se instanciará con el contenido recibido desde el servidor.

Tras ello, mediante un if, comprobaremos el contenido recibido sea “FIN_LISTA_ARCHIVOS” en cuyo caso instanciará a true la variable finArchivos y en caso contrario mostrará el contenido de la variable nombreFichero para finalmente realizar el break de este case.

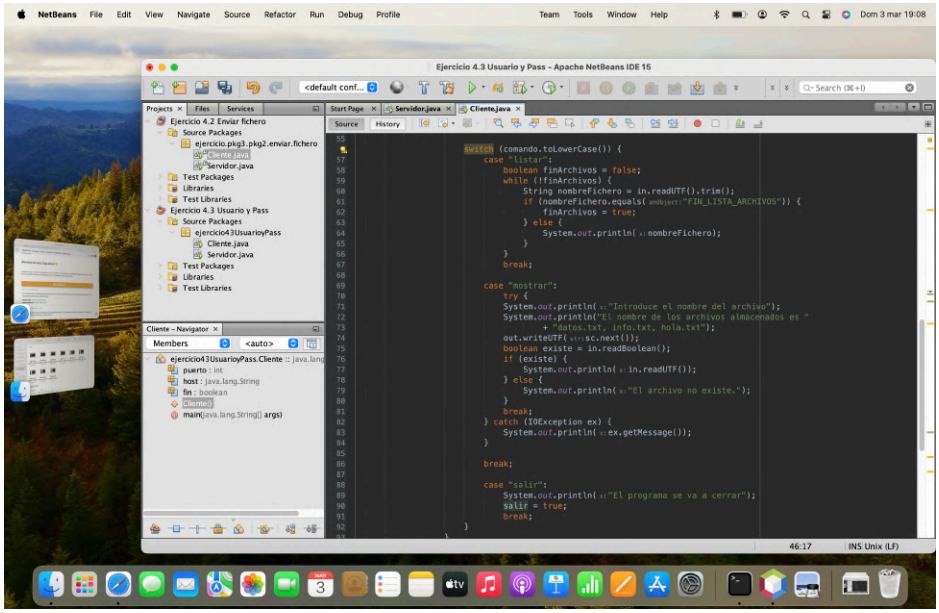
El case mostrar será el encarado de mostrar el contenido del archivo indicado por el cliente.

Mostrara al usuario un string para que introduzca el nombre del archivo, y otro string indicando los archivos disponibles.

Posteriormente enviará al servidor el nombre del archivo introducido por el cliente y creará e instanciará un boolean llamado existe con el contenido recibido desde el servidor.

Tras ello crearemos un if,, que comprobará si existe esta en true, en cuyo caso mostrara el contenido enviado desde el servidor.

En caso contrario indicara que el archivo no existe para posteriormente hacer un break de este case.

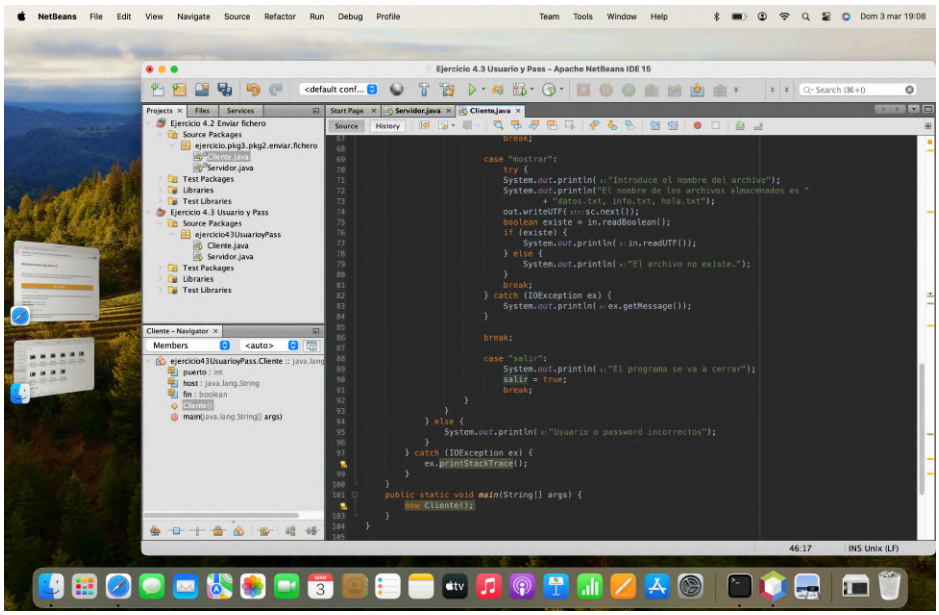


El case salir mostrara el string indicando que el programa se cerrará, cambiara el valor del boolean salir a true y hará su break.

En el caso de que el usuario no introduzca de forma correcta el usuario o la contraseña el programa se cerrará.

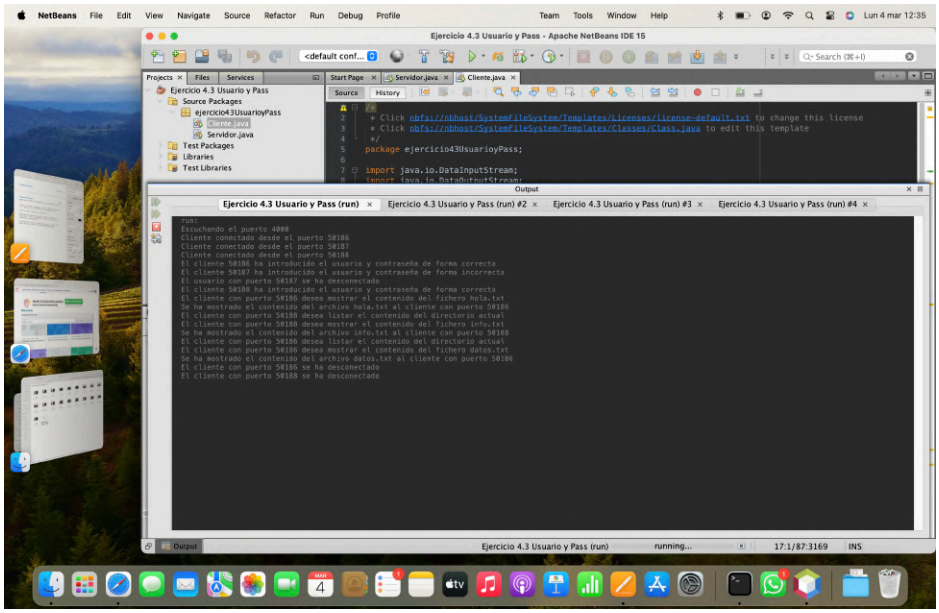
Todo ello estará envuelto en un try – catch para controlar las posibles excepciones.

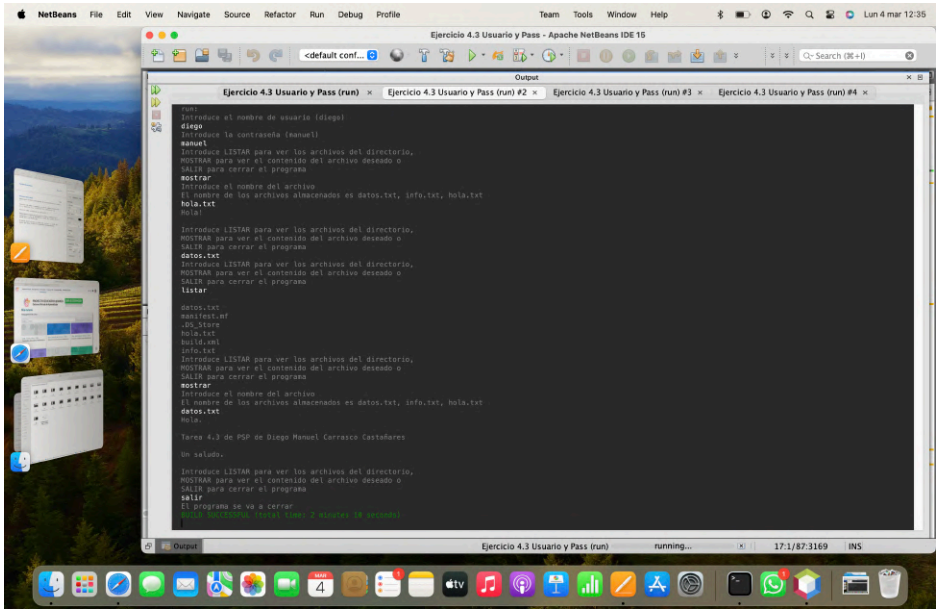
Por último constará del método main, que tendrá como única función llamar al constructor de esta clase.

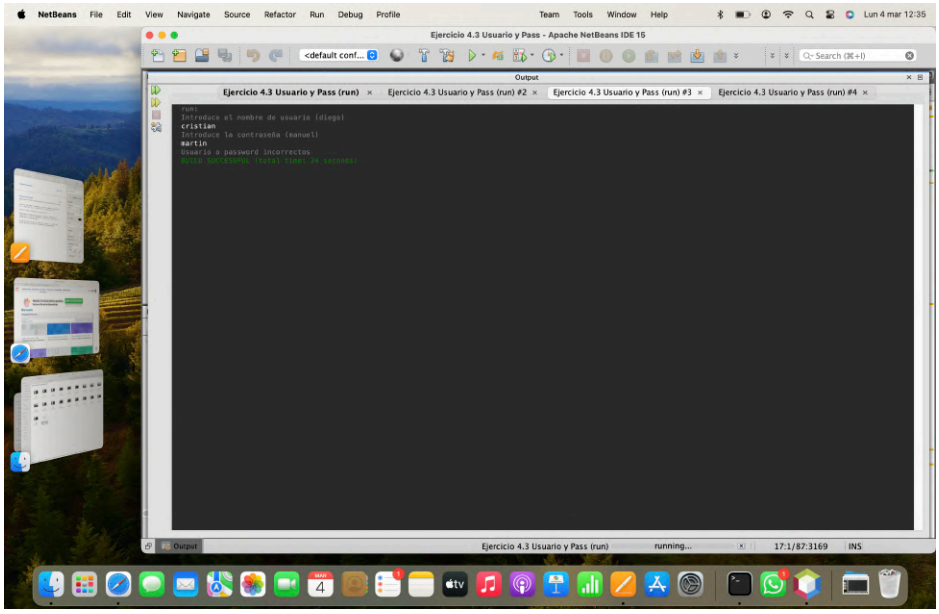


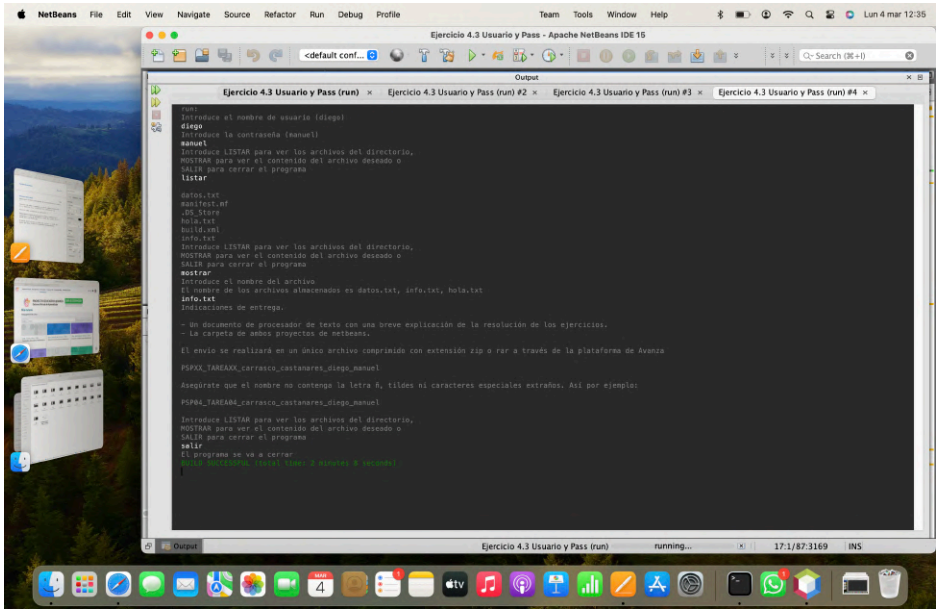
En las siguiente imágenes podemos ver los output de este ejercicio donde hemos iniciado una clase servidor y tres clases cliente, en dos de ellas he introducido de forma correcta el usuario y la contraseña y en otra lo he introducido de forma incorrecta para ver su comportamiento.

Y con las siguientes imágenes damos por finalizada esta tarea.




Ejercicio 4.3 Usuario y Pass - Apache NetBeans IDE 15
Ejercicio 4.3 Usuario y Pass (run)
Ejercicio 4.3 Usuario y Pass (run) #2
Ejercicio 4.3 Usuario y Pass (run) #3
Ejercicio 4.3 Usuario y Pass (run) #4
Introduce el nombre de usuario (diego)
diego
Introduce la contraseña (manuel)
manuel
Introduce LISTAR para ver los archivos del directorio,
MOSTRAR para ver el contenido del archivo deseado o
SALIR para cerrar el programa
mostrar
Introduce el nombre del archivo
El nombre de los archivos almacenados es datos.txt, info.txt, hola.txt
hola.txt
Hola!
Introduce LISTAR para ver los archivos del directorio,
MOSTRAR para ver el contenido del archivo deseado o
SALIR para cerrar el programa
datos.txt
Introduce LISTAR para ver los archivos del directorio,
MOSTRAR para ver el contenido del archivo deseado o
SALIR para cerrar el programa
listar
datos.txt
manifest.mf
.DS_Store
hola.txt
build.xml
info.txt
Introduce LISTAR para ver los archivos del directorio,
MOSTRAR para ver el contenido del archivo deseado o
SALIR para cerrar el programa
mostrar
Introduce el nombre del archivo
El nombre de los archivos almacenados es datos.txt, info.txt, hola.txt
datos.txt
Hola.
Tarea 4.3 de PSP de Diego Manuel Carrasco Castañares
Un saludo.
Introduce LISTAR para ver los archivos del directorio,
MOSTRAR para ver el contenido del archivo deseado o
SALIR para cerrar el programa
salir
El programa se va a cerrar
Ejercicio 4.3 Usuario y Pass (run)
running...
17:1/87:3169
INS


Ejercicio 4.3 Usuario y Pass - Apache NetBeans IDE 15
Ejercicio 4.3 Usuario y Pass (run)
Ejercicio 4.3 Usuario y Pass (run) #2
Ejercicio 4.3 Usuario y Pass (run) #3
Ejercicio 4.3 Usuario y Pass (run) #4
Introduce el nombre de usuario (diego)
cristian
Introduce la contraseña (manuel)
martin
Usuario o password incorrectos
Ejercicio 4.3 Usuario y Pass (run)
running...
17:1/87:3169
INS


Ejercicio 4.3 Usuario y Pass - Apache NetBeans IDE 15
Ejercicio 4.3 Usuario y Pass (run)
Ejercicio 4.3 Usuario y Pass (run) #2
Ejercicio 4.3 Usuario y Pass (run) #3
Ejercicio 4.3 Usuario y Pass (run) #4
Introduce el nombre de usuario (diego)
diego
Introduce la contraseña (manuel)
manuel
Introduce LISTAR para ver los archivos del directorio,
MOSTRAR para ver el contenido del archivo deseado o
SALIR para cerrar el programa
listar
datos.txt
manifest.mf
.DS_Store
hola.txt
build.xml
info.txt
Introduce LISTAR para ver los archivos del directorio,
MOSTRAR para ver el contenido del archivo deseado o
SALIR para cerrar el programa
mostrar
Introduce el nombre del archivo
El nombre de los archivos almacenados es datos.txt, info.txt, hola.txt
info.txt
Indicaciones de entrega.
- Un documento de procesador de texto con una breve explicación de la resolución de los ejercicios.
- La carpeta de ambos proyectos de netbeans.
El envío se realizará en un único archivo comprimido con extensión zip o rar a través de la plataforma de Avanza
PSPXX_TAREAXX_carrasco_castanares_diego_manuel
Asegúrate que el nombre no contenga la letra Ñ, tildes ni caracteres especiales extraños. Así por ejemplo:
PSP04_TAREA04_carrasco_castanares_diego_manuel
Introduce LISTAR para ver los archivos del directorio,
MOSTRAR para ver el contenido del archivo deseado o
SALIR para cerrar el programa
salir
El programa se va a cerrar
Ejercicio 4.3 Usuario y Pass (run)
running...
17:1/87:3169
INS