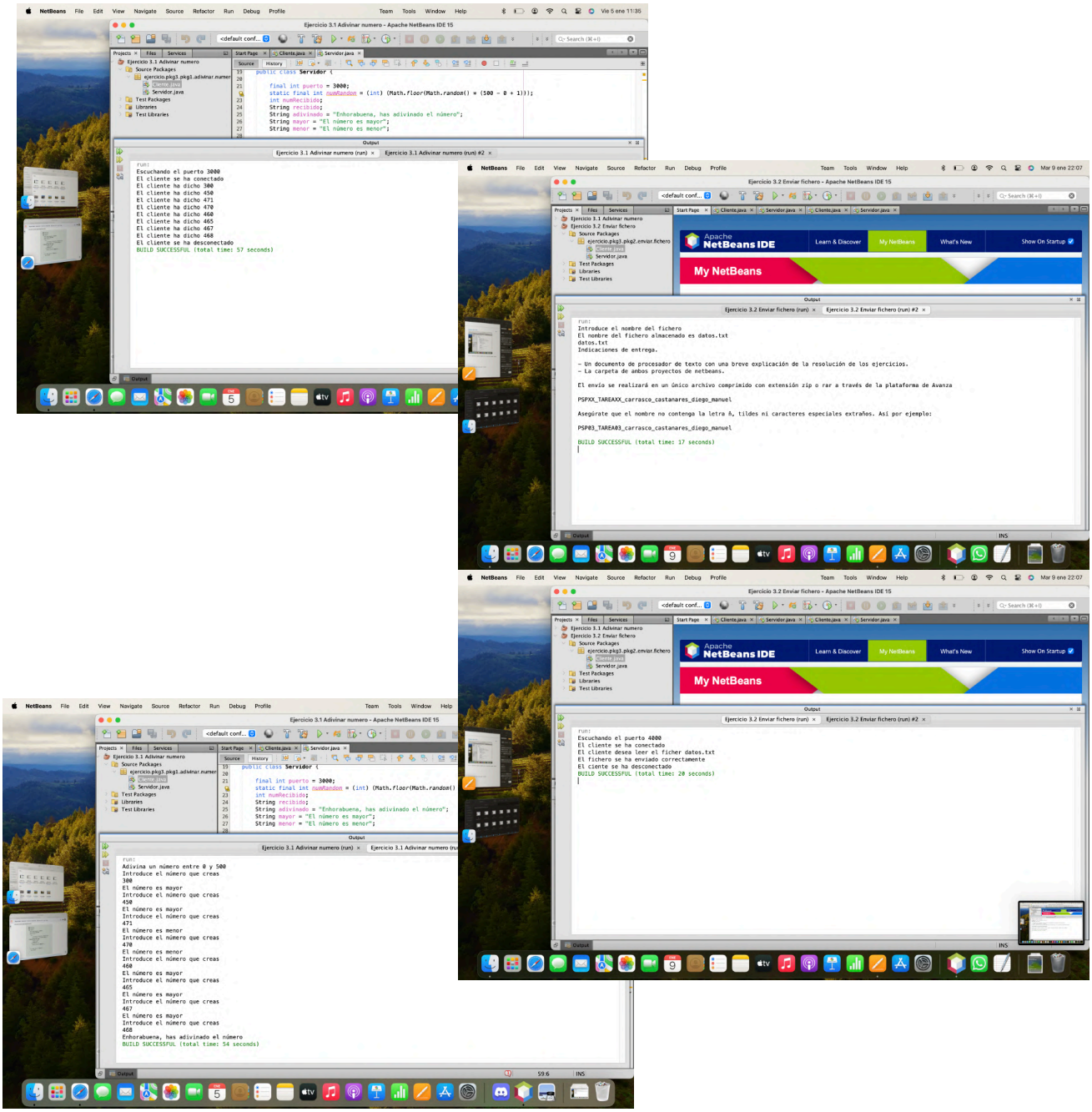
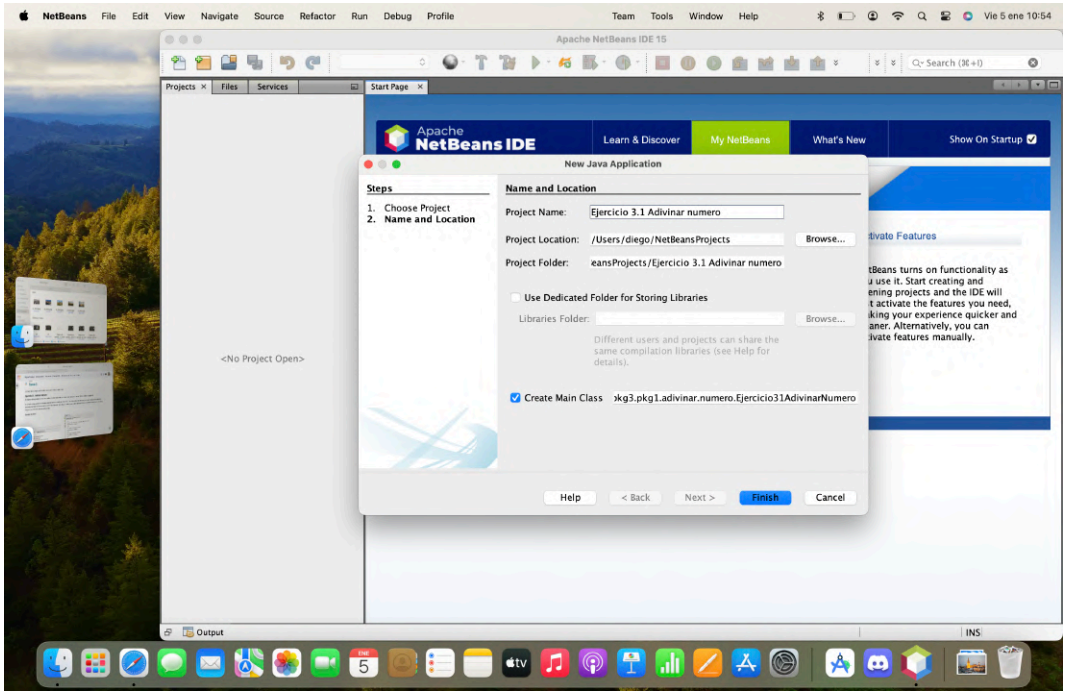
Tarea 3 para Programación de Servicios y Procesos

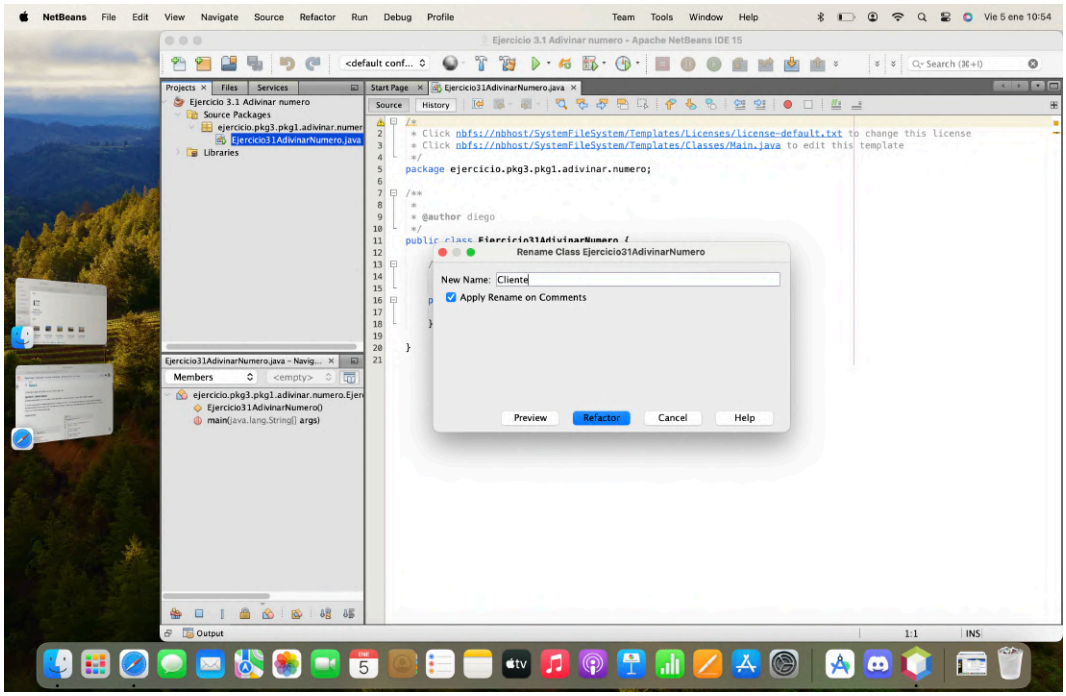


Diego Manuel Carrasco Castañares

En primer lugar, crearemos el proyecto en NetBeans, el cual se llamará, Ejercicio 3.1 Adivinar número.



Como lo he creado con main incluido voy a refactorizar el nombre de la clase, a la cual llamaré 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.

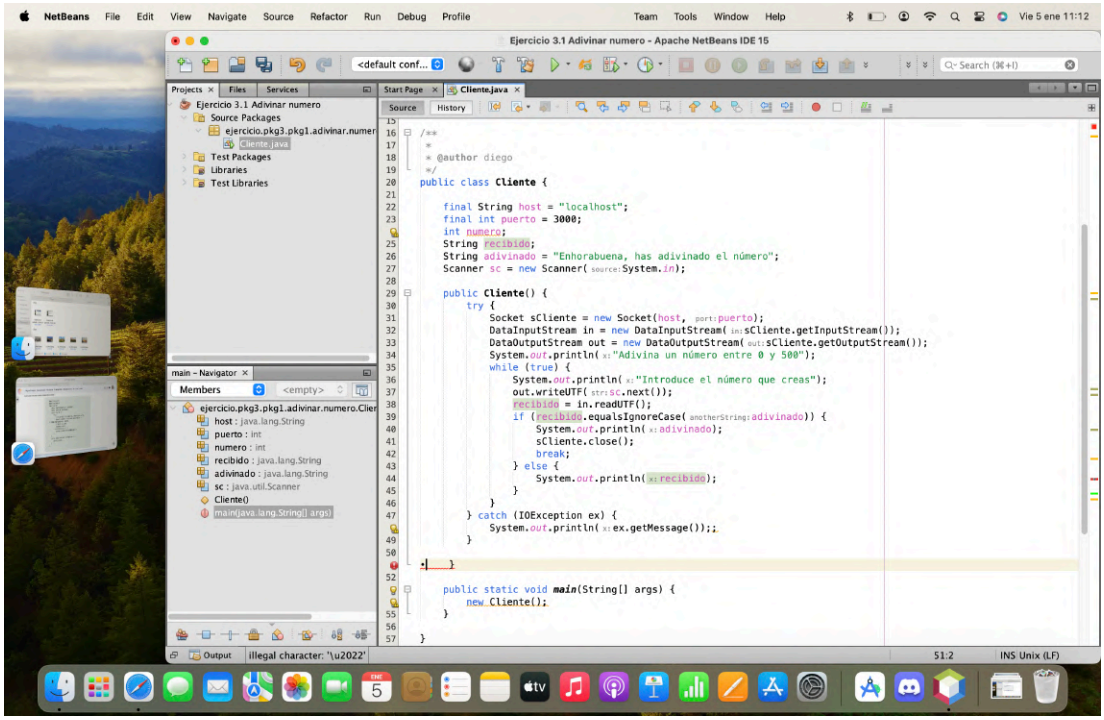
Tras ello crearemos un if que comprobará, mediante le método equalsIgnoreCase, si la variable recibido es igual a la variable adivinado y si es así mostrará el mensaje almacenado en adivinado al usuario y cerrara el socket sCliente.

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.

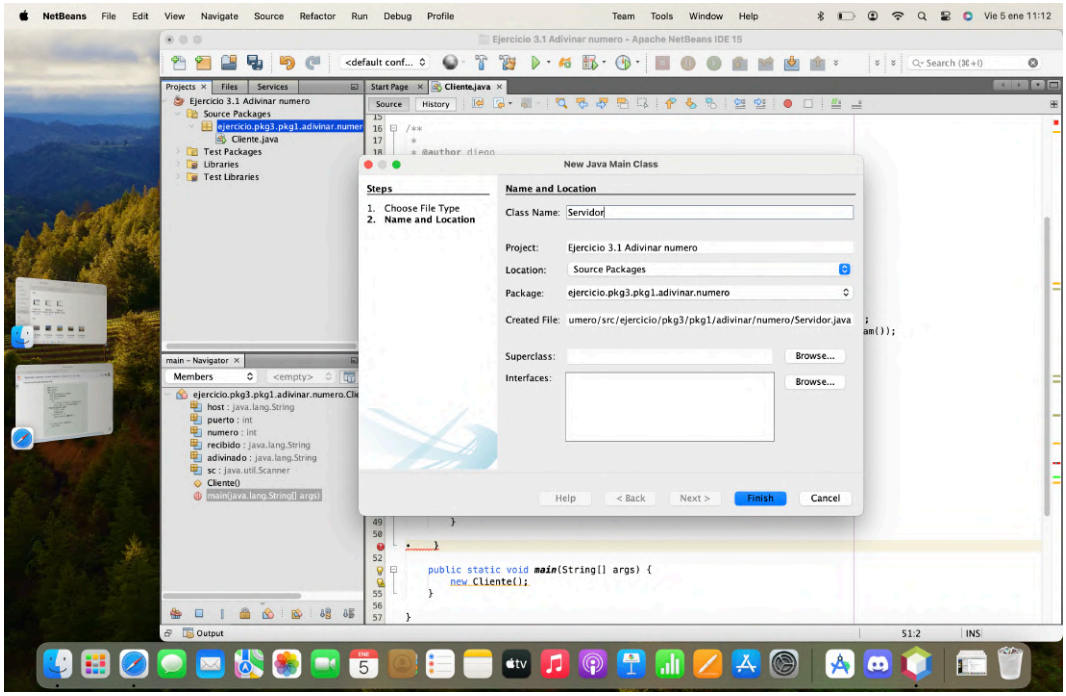
Todo esto estará envuelto en un try – catch.

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

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



El siguiente paso será crear la clase llamada servidor, la cual también dispondrá de método main.



En ella crearemos una variable tipo int y final llamada puerto, que contendrá el número de puerto, 3000, en este caso, tal y como indica la tarea.

Crearemos una variable de tipo int, estática y final llamada numRandom, que será instanciada con un número random entre el 1 y el 500 cada vez que se ejecute esta clase.

Crearemos otra variable de tipo int llamada numRecibido, que almacenara el número de tipo entero, que en nuestro caso almacenará el número recibido a través del socket después de pasarlo de string a integer mediante el método Integer.parseInt.

Crearemos una variable de tipo string llamada recibido, que almacenará el número recibido, en tipo string, desde el socket.

También crearemos tres variables mas de tipo string llamadas adivinado, mayor y menor, que contendrán los mensajes “Enhorabuena, has adivinado el número”, “El número es mayor” y “El número el menor” respectivamente.

Tras ello crearemos el método constructor.

Dicho método contendrá un ServerSocket llamado servidor, al cual le pasaremos el número de puerto.

Le mostraremos por pantalla al usuario el mensaje “Escuchando el puerto “ concatenado el número de puerto usado.

Crearemos un Socket llamado sCliente que aceptara el puerto usado por el servidor y una vez aceptado, mostrara por pantalla la usuario el mensaje “El cliente se ha conectado”.

Tras ello crearemos un DataInputStream llamado in, el cual será instanciado con el contenido del socket mediante el método getInputStream y un DataOutputStream llamado out que instanciara el el socket sCliente lo contenido en el out mediante el método dataOutputStream.

Tras ello crearemos un while con la condición verdadero.

En dicho while leeremos, mediante el método readUTF el contenido de la variable recibido, mostrara por pantalla el mensaje “El cliente ha dicho “ concatenado con la cadena contenida en dicha variable y pasará esa cadena a entero mediante el método Integer.parseInt.

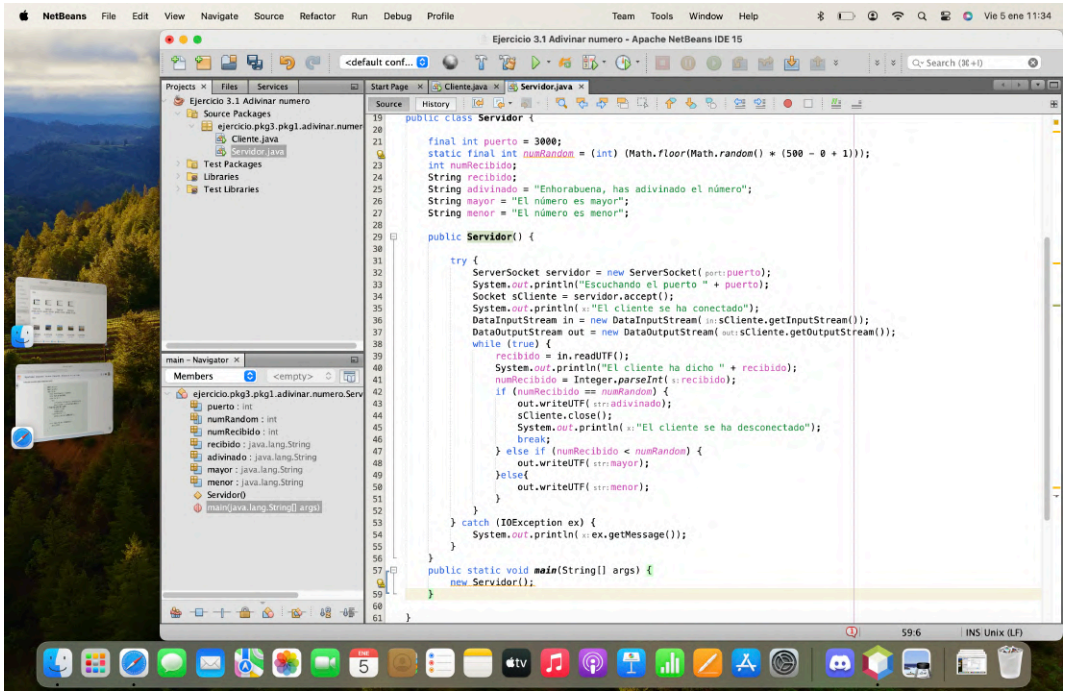
Tras ello crearemos un if que comprobara si el número almacenado en la variable numRecibido es igual al número almacenado en la variable numRandom, y si es así instanciara en la variable out el mensaje contenido en la variable adivinado, cerrara el socket sCliente y mostrara por pantalla el mensaje “El cliente se ha desconectado”.

Mediante un else -if, y si el número almacenado en numRecibido es menor que el número almacenado en numRandom instanciara en el out el mensaje almacenado en la variable mayor.

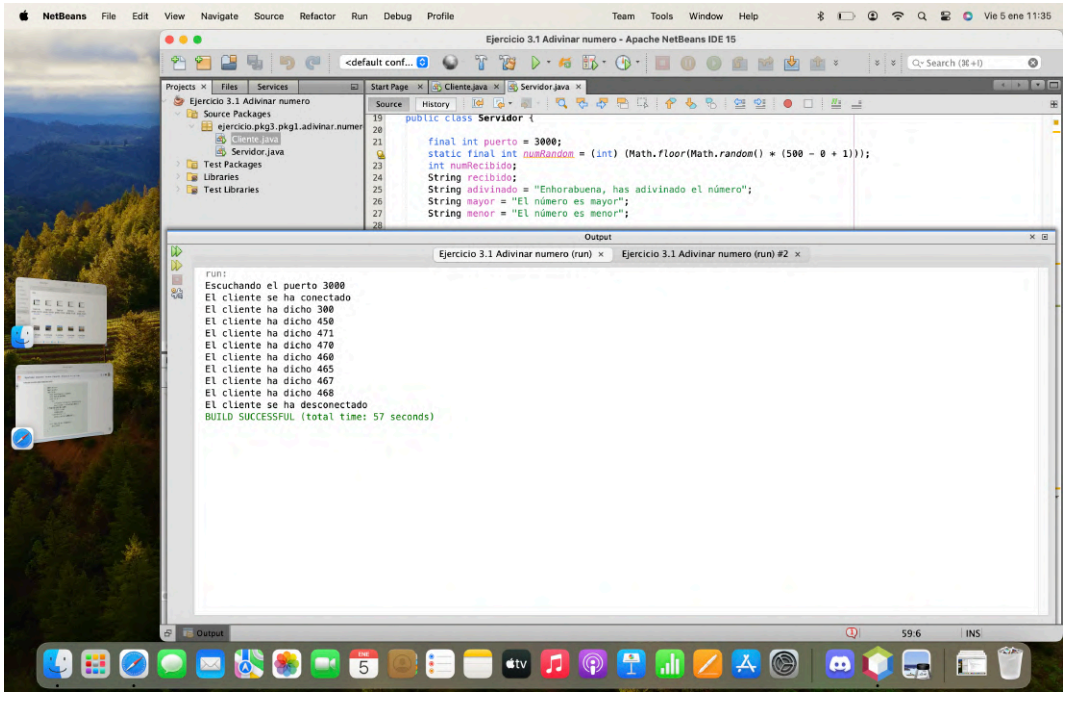
Por el contrario, si no se cumplen ninguna de las dos condiciones anteriores, instanciara en la variable out el mensaje almacenado en la variable menor.

Todo lo descrito anteriormente, creado en el método constructor, estará envuelto en un try – catch.

Por último crearemos el método main, en el cual llamaremos al método constructor de esta clase.

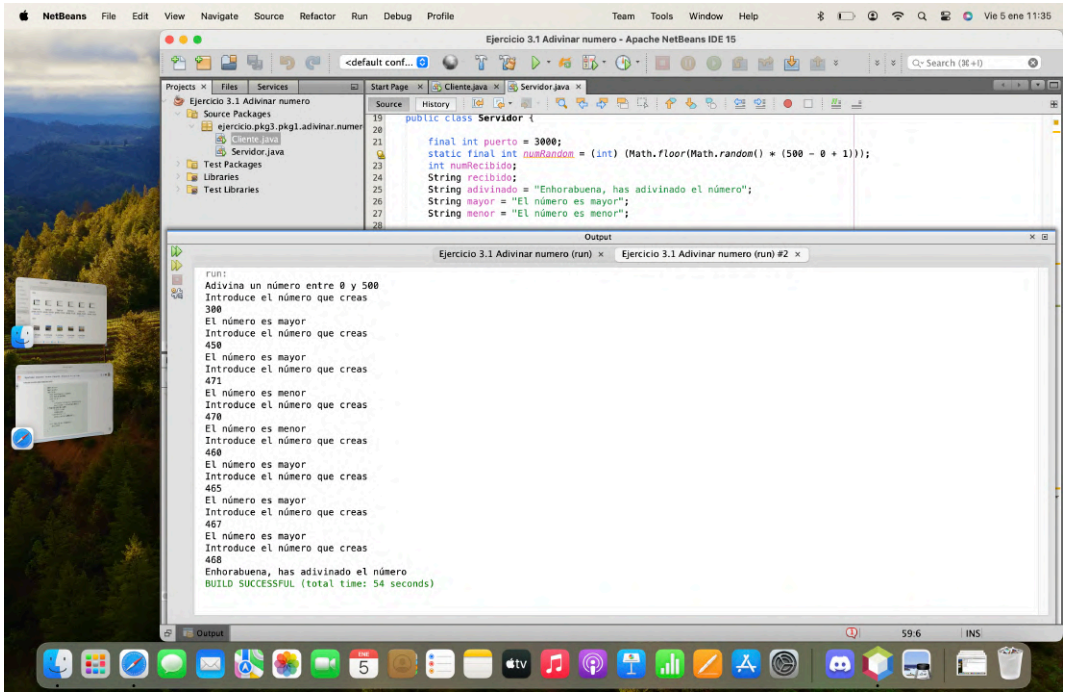


En las siguiente imagen podemos ver la salida por consola de la case servidor desde que se ha ejecutado hasta que ha sido adivinado el número por parte del usuario. Podemos ver como al adivinar el número la ejecución del programa termina.

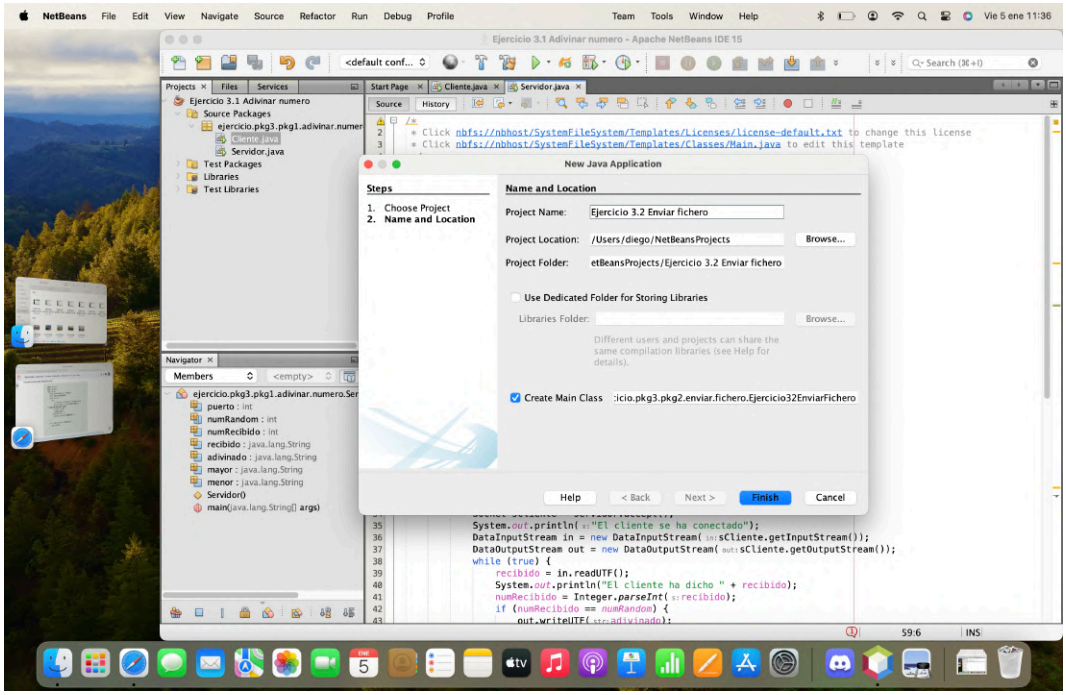


En la siguiente imagen podemos ver la salida por consola de la clase cliente desde que ha sido ejecutada hasta que el usuario ha adivinado el número.

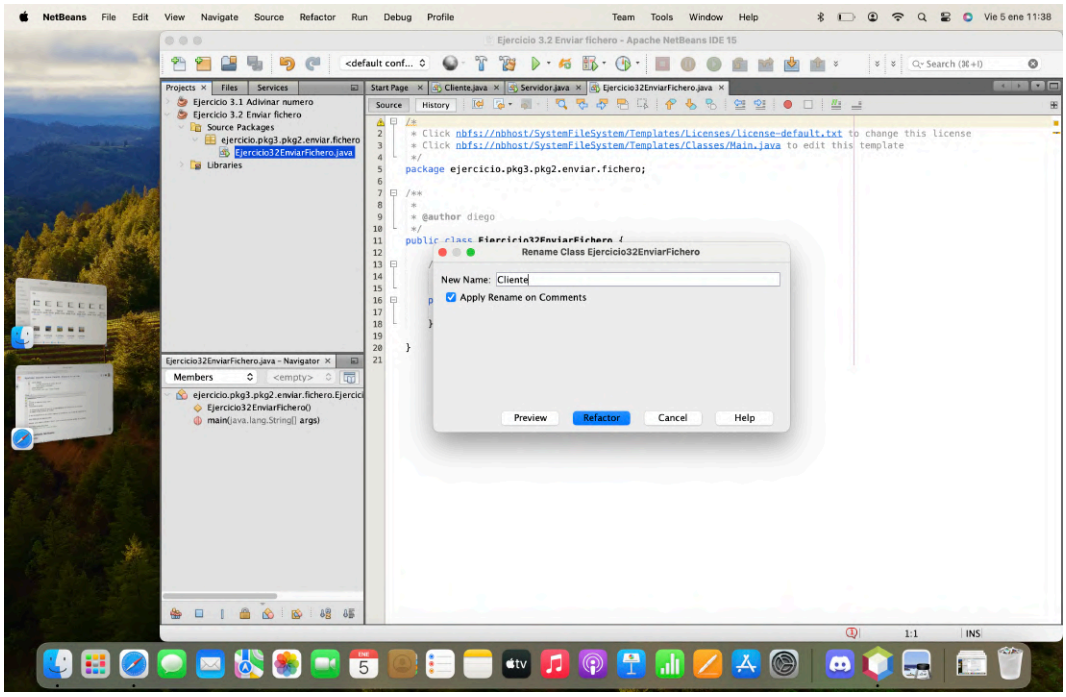
Podemos ver, al igual que en la foto anterior, como al adivinar el número la ejecución del programa termina.



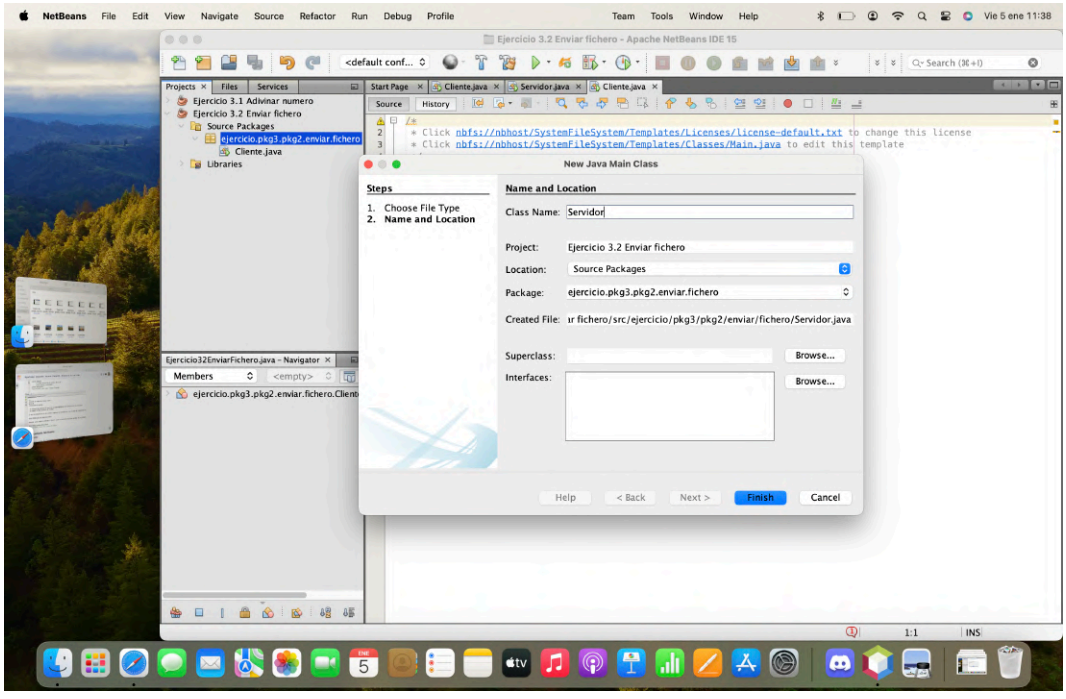
Con esto damos por finalizada esta parte de la tarea.
Ahora crearemos un nuevo proyecto en NetBeans llamado Ejercicio 3.2 Enviar fichero.



Como también lo he creado con método main, voy a refactorizar el nombre de la clase a la cual llamaré cliente.



Posteriormente crearemos la clase servidor, así ya la dejamos creada.



Volviendo a la clase cliente, 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 DataOutputStreama 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.

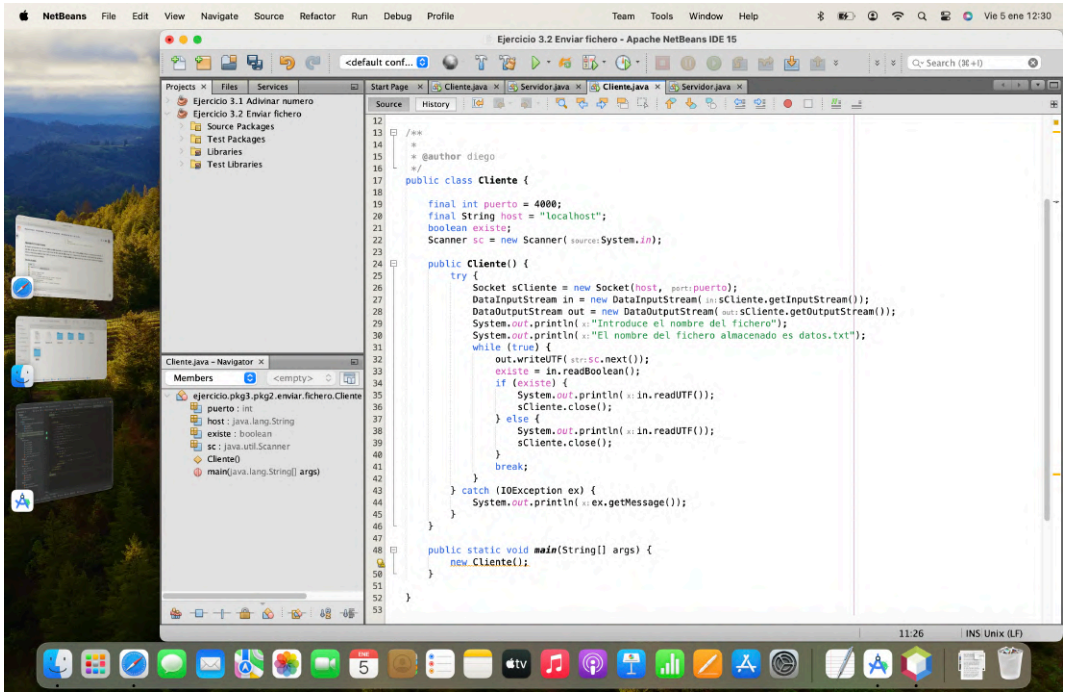
Lo siguiente será crear un if, cuya condición será la variable existe, que si es verdadera, mostraremos por pantalla el contenido de la variable in aplicándole el método readUTF y cerraremos el socket sCliente.

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.

En la siguiente imagen podemos ver el código la clase mencionada



Volviendo a la clase servidor, empezaremos creando las variables.

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

Las demás variables serán de tipo string, empezando pro una llamada nomFichero, que almacenara el nombre del fichero, otra llamada mensaje que contendrá la cadena “El fichero mencionado no existe”, otra llamada línea, que almacenara cada línea del fichero leía y la última llamada documento, que contendrá todo el texto que incluye el documento.

El siguiente paso será crear el método constructor.

Dentro de él crearemos un ServerSocket llamado servidor al cual le pasaremos la variable puerto.

Tras ello mostraremos por pantalla el mensaje “Escuchando puerto “ concatenado con la variable puerto.

A continuación crearemos un socket llamado sCliente al cual le pasaremos el puerto que tiene el servidor mediante el método accept.

Tras ellos mostraremos por pantalla el mensaje “El cliente se ha conectado”.

Tras ello crearemos un DataInputStream y un DataOutputStream que se llamaran in y out respectivamente y serán instanciados con el contenido del socket sCliente a través de los métodos getInputStream y getOutputStream respectivamente.

A continuación instanciaremos en la variable nomFichero el contenido de la variable in mediante el método readUTF.

El siguiente paso será mostrar por pantalla el mensaje “El cliente desea leer el fichero “ concatenado con la variable nomFichero.

Tras ello crearemos una variable de tipo fichero llamada fichero a la cual le pasaremos la variable nomFichero como pathname.

A continuación crearemos un if, que comprobara si el fichero existe, y si es así, instanciara en true la variable in mediante el método writeBoolean, crearemos un FileReader llamado fr pasándole el fichero y crearemos un BufferedReader llamado br al cual le pasaremos la variable fr.

Tras ello crearemos un while, que mientras que haya líneas que leer en el fichero, añadirá dicha linea al documento y hará un salto de linea.

A continuación instanciaremos en la variable out, mediante el método wetiteUTF, el contenido de la variable documento.

El siguiente paso será cerraer el socket sCliente y el bufferedReader.

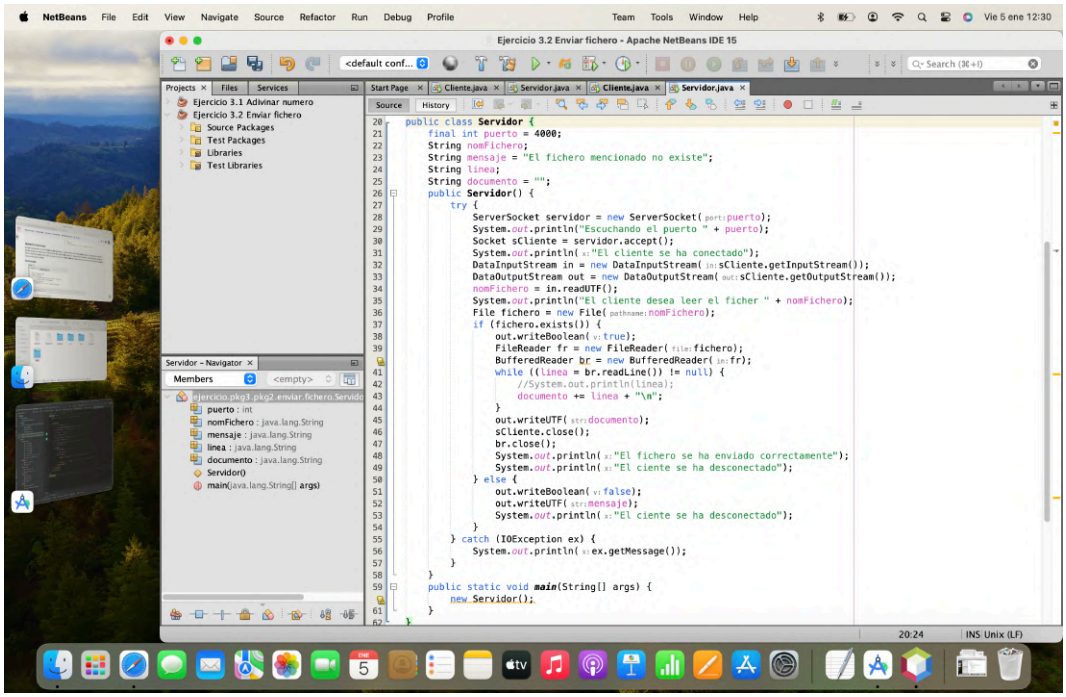
Tras ello mostraremos por pantalla los mensajes “El fichero se ha enviado correctamente” y “El cliente se ha desconectado”.

En caso contrario (del if creado anteriormente) instanciaremos false en la variable out mediante el método writeBoolean, instanciaremos en la misma variable out la cadena contenida en la variable mensaje y mostraremos por pantalla el mensaje “El cliente se ha desconectado.

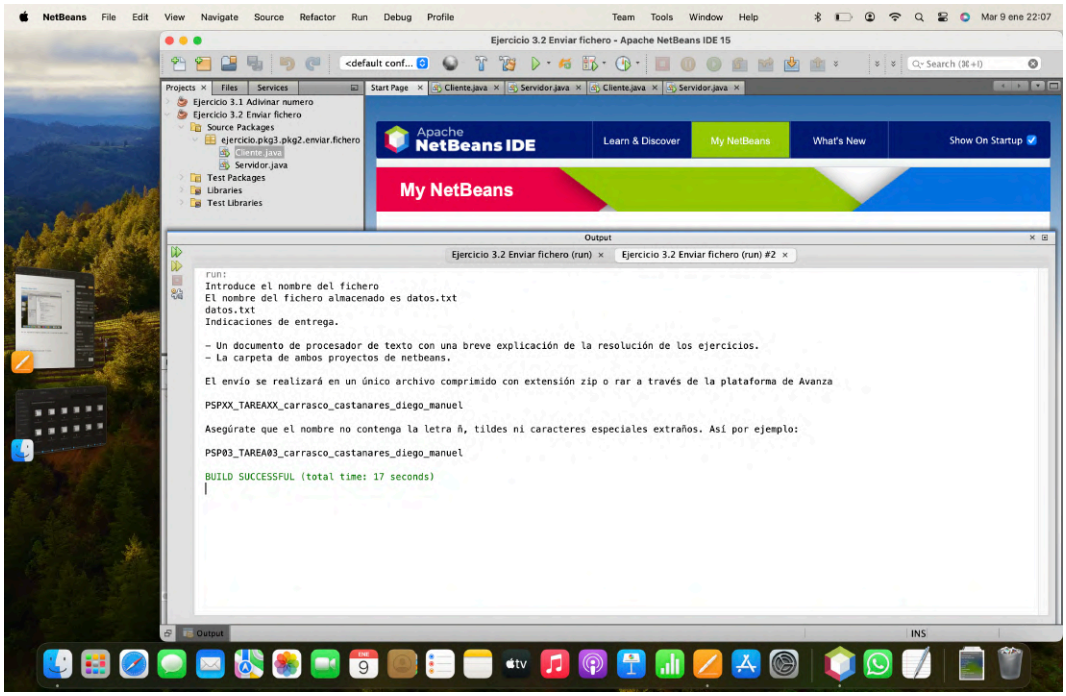
Todo el código incluido en el constructor estará envuelto en un try – catch.

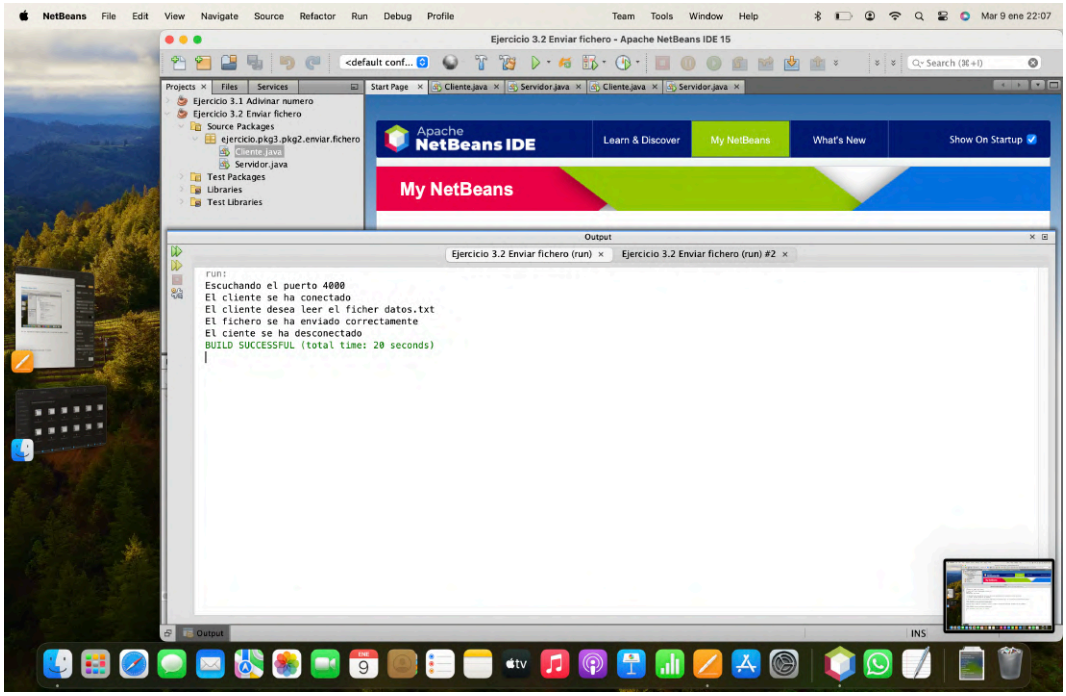
Por último crearemos el main, dentro del cual llamaremos al método constructor de la clase.

En la siguiente imagen podemos ver el código de esta clase.



En las siguientes imágenes podemos ver la salida de ambas clases.





Y con esto damos por finalizada la tarea.