Programación de

Servicios y Procesos

CLIENTE-SERVIDOR

Diego González Martínez | 26/12/2022

# Ejercicio 3.1

**¿De qué sirve el Ejercicio 3?**

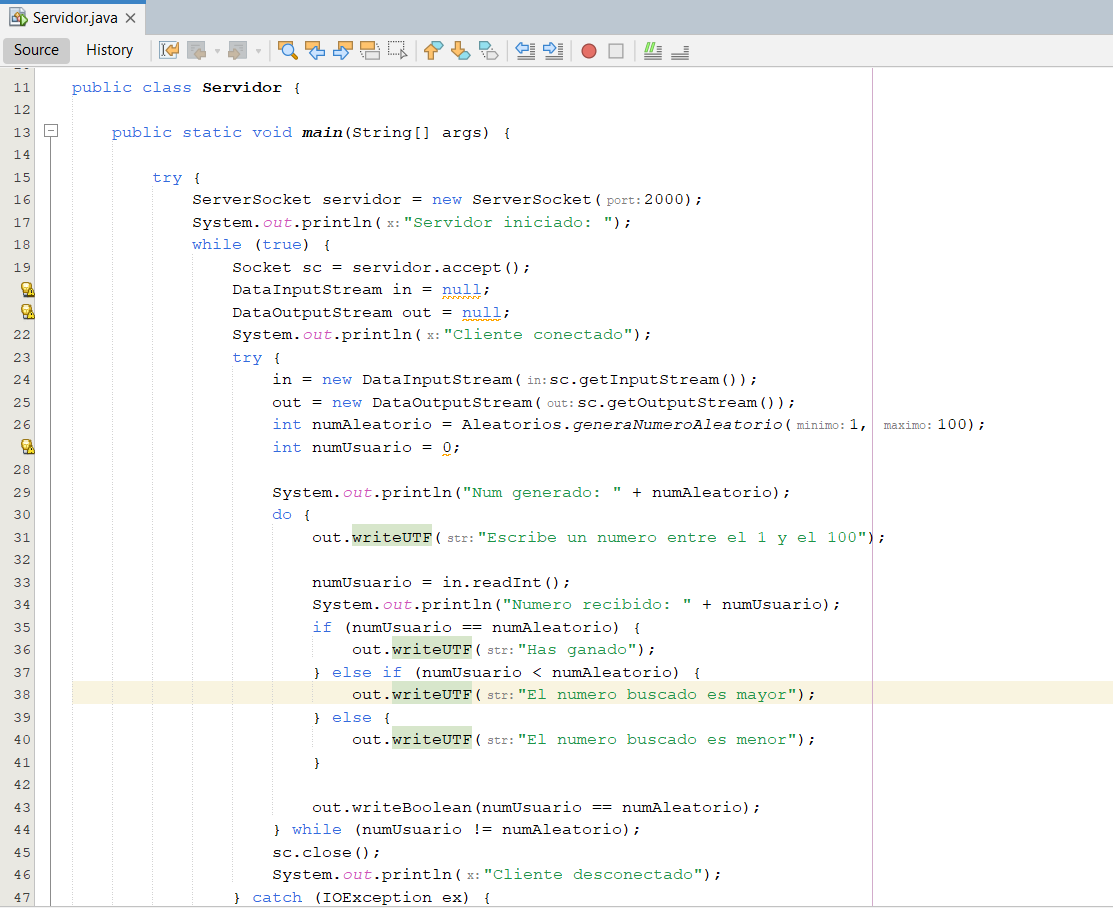
**Actividad 3.1.** El objetivo del ejercicio es crear una aplicación cliente/servidor que se comunique por el puerto 2000 y realice lo siguiente:

El servidor debe generar un número secreto de forma aleatoria entre el 0 al 100. El objetivo de cliente es solicitarle al usuario un número y enviarlo al servidor hasta que adivine el número secreto. Para ello, el servidor para cada número que le envía el cliente le indicará si es menor, mayor o es el número secreto del servidor.

**La lógica del código es la siguiente:**

1. Creo un ServerSocket en el puerto 1500, que escucha las conexiones entrantes.
2. Una vez que se establece una conexión, el programa lee la ruta del archivo que es enviada por el cliente.
3. Luego, utilizo la clase File para verificar si el archivo existe.
4. Si el archivo existe, el programa utiliza un BufferedReader para leer el contenido del archivo y almacena el contenido en una variable de tipo string.
5. Luego, convierto la variable de tipo string a un arreglo de bytes y envía el arreglo de bytes al cliente utilizando el DataOutputStream.
6. Si el archivo no existe, envío un valor booleano falso al cliente.
7. El programa utiliza un ciclo while para mantener el servidor ejecutándose de forma continua y un bloque try-catch para manejar cualquier excepción de tipo IOException que pueda ocurrir.
8. Principio del formulario

**Clase Servidor:**



La clase **Servidor** tiene un método **main** que se ejecuta cuando se ejecuta el programa. Dentro del método, se crea un **ServerSocket** en el puerto 2000. El servidor imprime "Servidor iniciado" y entra en un bucle infinito, aceptando conexiones entrantes. Cada vez que un cliente se conecta, se crea un **Socket** para la conexión y se inicializan **DataInputStream** y **DataOutputStream** para comunicarse con el cliente. El servidor genera un número aleatorio entre 1 y 100 utilizando un método **generaNumeroAleatorio** en la clase **Aleatorios**.

El código entra en un bucle que se ejecuta mientras el número adivinado por el cliente sea incorrecto. Dentro del bucle, el servidor envía un mensaje al cliente pidiendo que adivine un número entre 1 y 100. El cliente envía un número de vuelta, el servidor lo recibe, y entonces verifica si es correcto comparando con el número aleatorio generado al principio. Si no es correcto, el servidor envía al cliente un mensaje indicando si el número adivinado es mayor o menor al número buscado. Si es correcto, el servidor envia el mensaje "Has ganado" y sale del bucle.

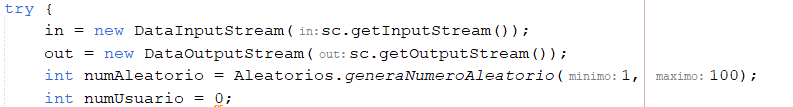
Cuando el juego termina, el servidor cierra la conexión con el cliente e imprime "Cliente desconectado" en la consola. Si ocurre algún error, se imprime "Error" en la consola.

**Explicación del código:**

****

**ServerSocket servidor = new ServerSocket(2000);** Crea una instancia de **ServerSocket** en el puerto 2000. El servidor escuchará en este puerto específico para conexiones entrantes.

**Socket sc = servidor.accept();** Acepta una nueva conexión entrante y crea un nuevo socket para la conexión.



**in = new DataInputStream(sc.getInputStream());** **out = new DataOutputStream(sc.getOutputStream());** Crea una nueva instancia de **DataInputStream** y **DataOutputStream** usando el **InputStream** y **OutputStream** del socket. Esto permite la lectura y escritura de datos a través de la conexión.



**int numAleatorio = Aleatorios.generaNumeroAleatorio(1, 100);** Crea un numero aleatorio entre 1 y 100. El metodo generaNumeroAleatorio es un metodo estatico de una clase llamada Aleatorios,



**out.writeBoolean(numUsuario == numAleatorio);** Envia un booleano al cliente indicando si el número adivinado es correcto o no.



**} while (numUsuario != numAleatorio);** Verifica si el número adivinado es correcto, si no es correcto, el bucle se ejecuta de nuevo.

**sc.close();** Cierra la conexión socket con el cliente.

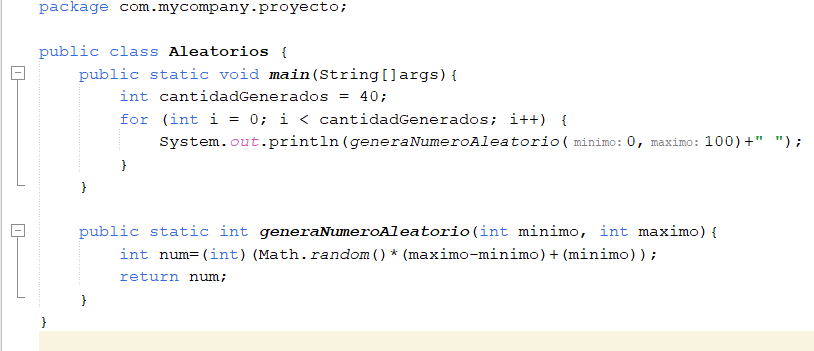
**Clase Cliente:**



Este código es el método main de una clase llamada "Cliente". El código crea una nueva conexión socket con un host especificado como "localhost" y un puerto 2000. La conexión establecida permite la comunicación entre el cliente y el servidor a través de dos flujos de datos: uno de entrada y otro de salida.

Entonces se declara un bucle "do-while" que se ejecutará hasta que la variable "salir" sea verdadera. En cada iteración del bucle, se recibe un mensaje en formato UTF a través del flujo de entrada y se imprime en la consola. Luego se espera una entrada del usuario, que se recoge con el objeto "Scanner" y se envía al servidor a través del flujo de salida. Finalmente, se recibe otro mensaje desde el servidor y se imprime en la consola, y se recibe un boolean indicando si el bucle debe continuar o no. En caso de que ocurra una excepción de E/S, el controlador captura y registra la excepción mediante el uso de la clase Logger. Finalmente, al terminar el bucle, cierra el socket.

**Clase Aleatorios:**



Este código Java es un programa que genera números aleatorios entre un valor mínimo y un valor máximo. El método principal de la clase Aleatorios primero declara una variable "cantidadGenerados" y le asigna el valor de 40. Luego, utiliza un ciclo for para iterar 40 veces y cada vez llama al método "generaNumeroAleatorio" con el valor mínimo de 0 y el valor máximo de 100. El método "generaNumeroAleatorio" utiliza el método Math.random() para generar un número aleatorio entre los valores mínimo y máximo proporcionados y devuelve ese número. Luego, el método principal imprime ese número seguido de un espacio. Este programa generará e imprimirá 40 números aleatorios entre 0 y 100.

JAVADOC:

C:\3.1Proyecto\target\site\apidocs

