Programación de

Servicios y Procesos

CLIENTE-SERVIDOR

Diego González Martínez | 03/01/2023

# Ejercicio 4.1

**¿De qué trata el Ejercicio 4?**

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

**La lógica del juego es el siguiente:**

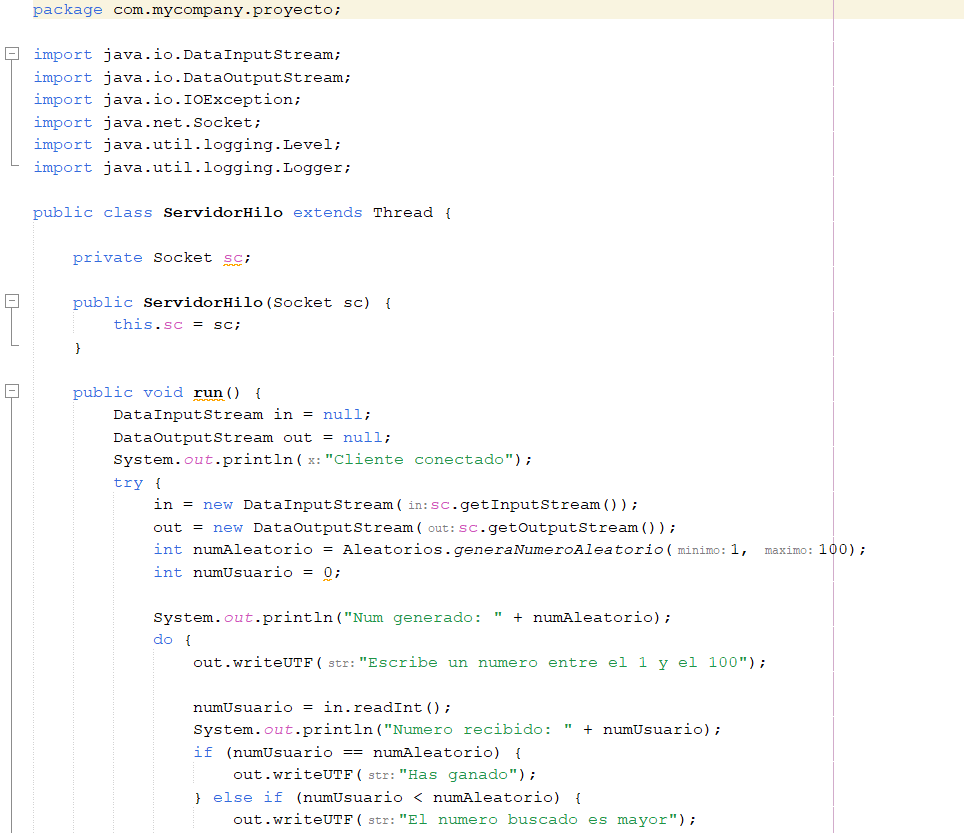
1. El servidor genera un número aleatorio entre 1 y 100 utilizando la clase "Aleatorios".
2. El servidor envía al cliente un mensaje pidiendo que escriba un número entre 1 y 100.
3. El cliente envía un número al servidor.
4. El servidor compara ese número con el número aleatorio generado anteriormente.
5. Si los números son iguales, el servidor envía al cliente un mensaje diciendo "Has ganado". Si el número enviado por el cliente es menor que el número aleatorio, el servidor envía un mensaje diciendo "El número buscado es mayor". Si el número enviado por el cliente es mayor que el número aleatorio, el servidor envía un mensaje diciendo "El número buscado es menor".
6. El bucle se repite hasta que el número enviado por el cliente sea igual al número aleatorio generado.
7. Por último el socket se cierra y el cliente se desconecta.

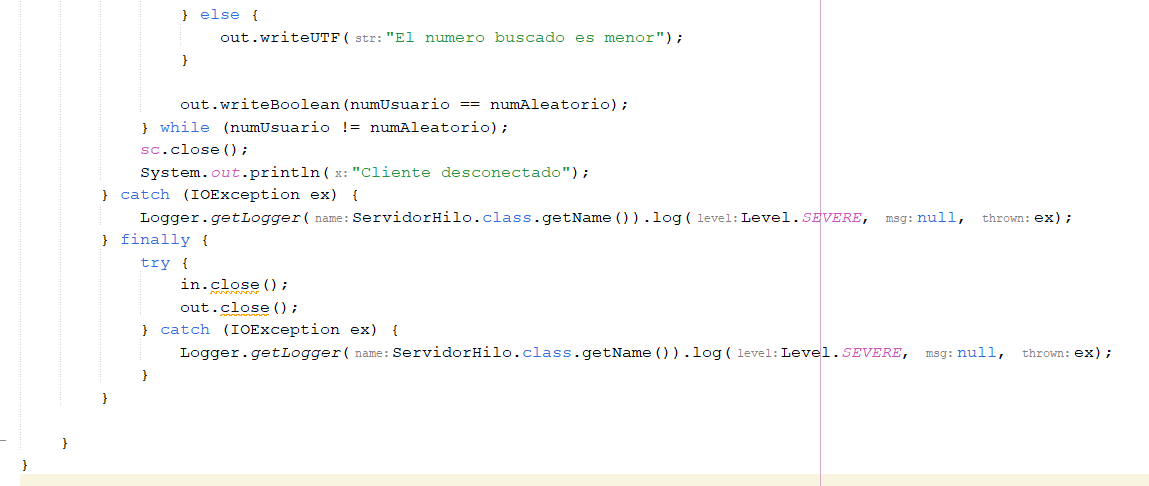
Es importante notar que, cuando se detecta alguna excepción en el tratamiento de los flujos de datos, se registra en un archivo de log.

**¿En qué se diferencia con el primer ejercicio 3.1?**

La diferencia con el primer codigo es que este utiliza una clase ServerSocket para crear un servidor que acepta varios clientes y maneja un bucle infinito para aceptar conexiones continuas y procesar los jugadores, mientras que el primero utilizaba Threads para manejar varios clientes de manera concurrente.

**Clase Servidor-Hilos**

****

****

Este código es una clase de Java llamada "ServidorHilo" que extiende de la clase Thread. Esta clase tiene un constructor que acepta un socket y en el método "run()" tiene lógica para interactuar con un cliente a través de ese socket.

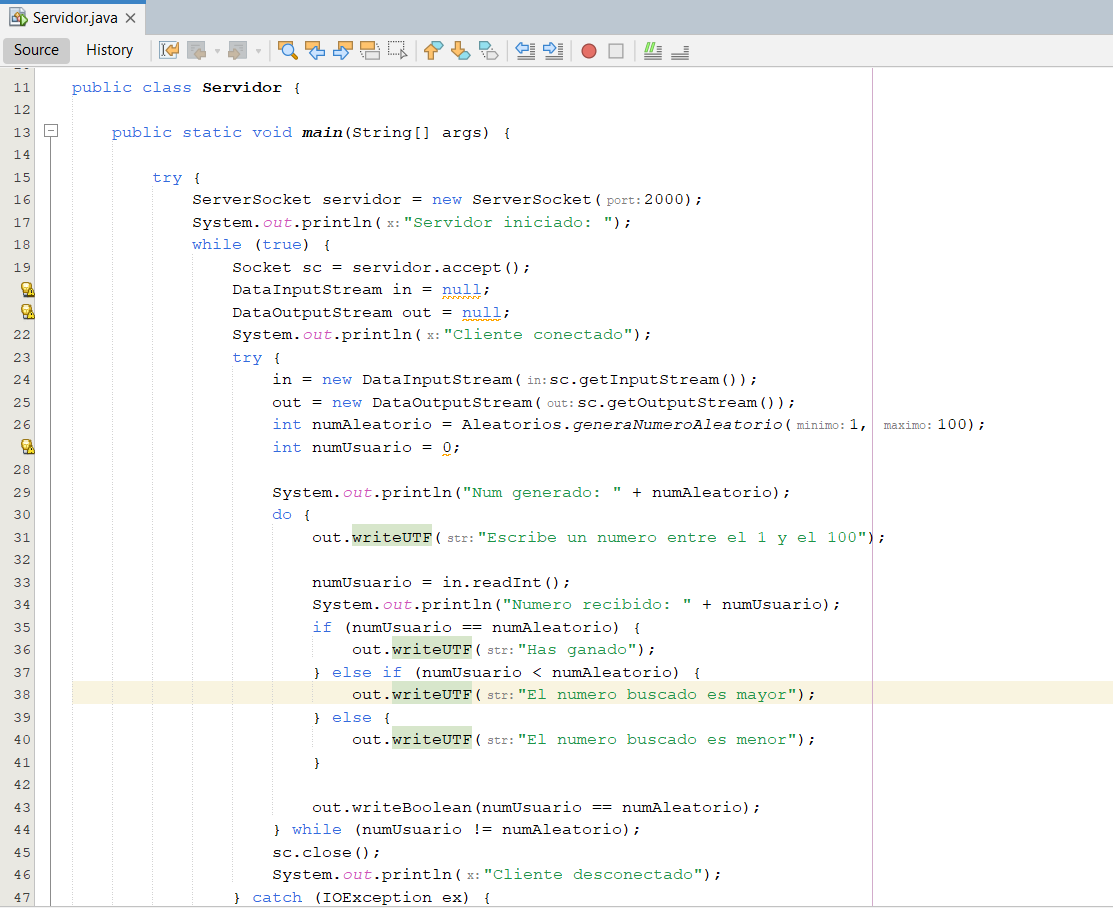
El servidor utiliza un socket para comunicarse con el cliente y utiliza dos flujos de datos, uno para enviar mensajes al cliente y otro para recibir números del cliente.

La clase se ejecuta en un hilo independiente para poder atender a varios clientes de manera concurrente. El juego se repite en bucle hasta que el cliente adivine el número generado por el servidor, entonces el servidor cierra el socket y termina la comunicación con el cliente.

En caso de que se produzca una excepción en el tratamiento de los flujos de datos, se captura y se registra en un archivo de log, esto permite mantener un control de los errores que ocurren en la aplicación.

Dentro de la clase ServidorHilo, el método run() genera un número aleatorio entre 1 y 100, y luego comienza un bucle. En cada iteración del bucle, el servidor envía un mensaje al cliente pidiéndole que ingrese un número, luego recibe el número ingresado por el cliente. Si el número ingresado es igual al número aleatorio generado, se envía un mensaje al cliente indicando que ha ganado, de lo contrario, se envía un mensaje indicando si el número buscado es mayor o menor. El bucle continúa hasta que el cliente adivine correctamente el número.

****



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.

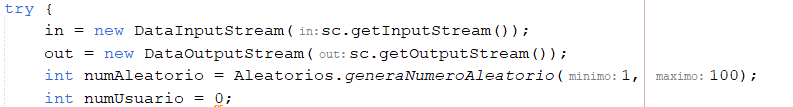
Cuando el juego termina, el servidor cierra la conexión con el cliente e imprime "Cliente desconectado" en la consola. Si ocurre algún error, se imprime "Error" en la consola.

**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.

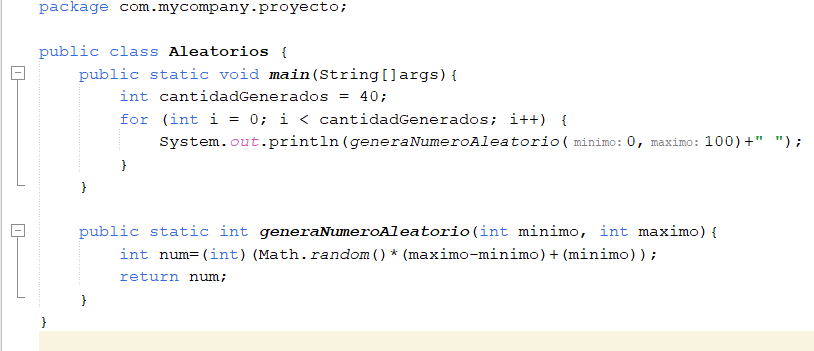
**Clase Cliente:**



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.

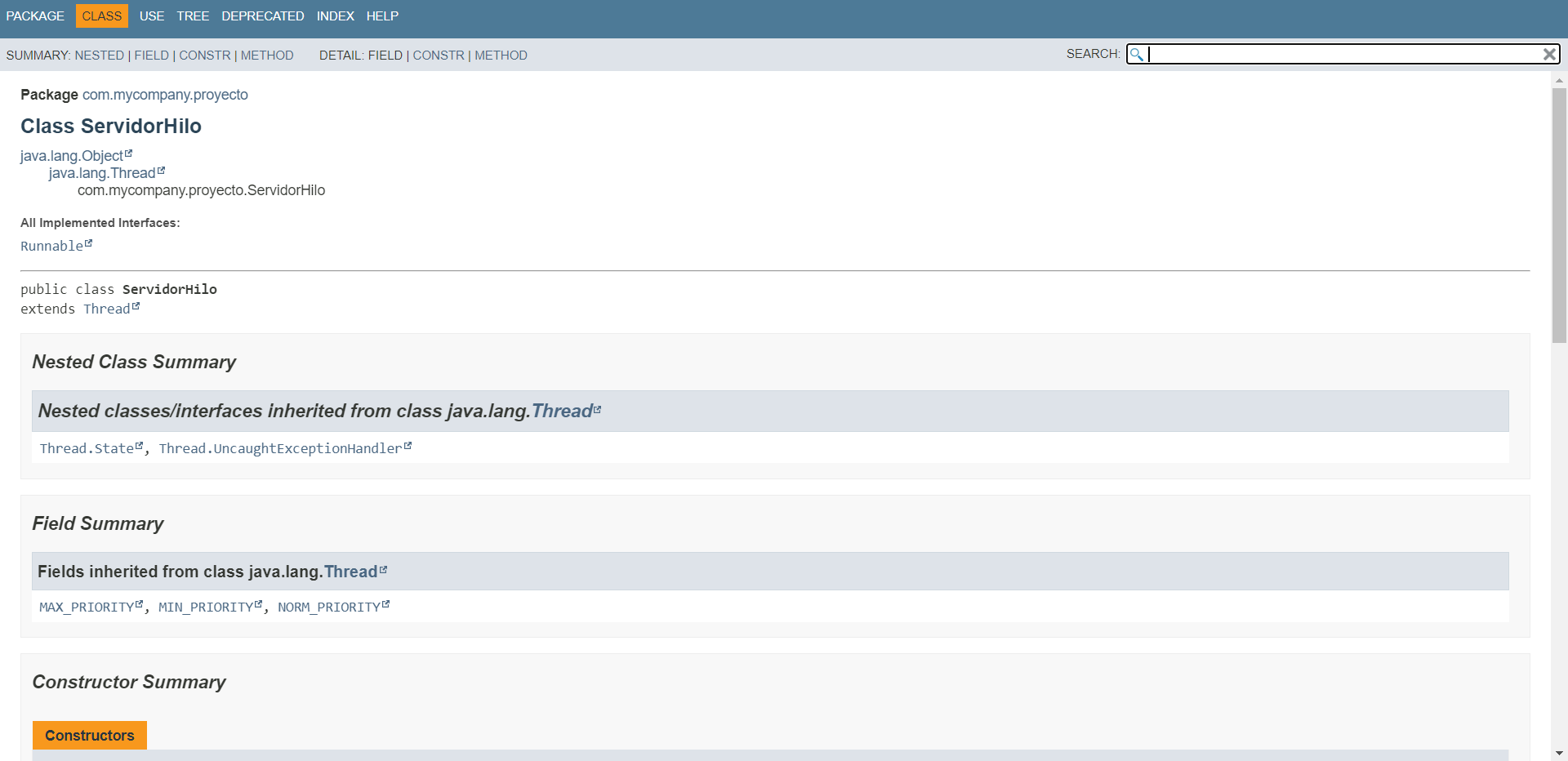
**Clase Aleatorios:**



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:\4.1Proyecto\target\site\apidocs**

****