UNIDAD 2: PROGRAMACION EN RED

RELACIÓN DE EJERCICIOS

1.- Tenemos que implementar, con Sockets, un modelo cliente-servidor único (es decir, un solo cliente y un único servidor, no se admiten conexiones múltiples, ni secuencial ni concurrente).

Una vez realiza la conexión, el cliente tendrá que adivinar un número que el servidor ha generado de forma aleatoria (entre 1 y 100).

De esta forma el cliente pedirá un número al usuario, lo enviará al servidor y este le responderá si es correcto, demasiado pequeño y demasiado grande (mensaje que se mostrará por pantalla).

Cuando se haya adivinado el número ambos procesos se cerrarán.

Ejemplo de ejecución de un servidor:

```
Esperando al cliente...
Recibiendo del cliente:
	50
Recibiendo del cliente:
	80
Recibiendo del cliente:
	60
Recibiendo del cliente:
	65
Recibiendo del cliente:
	70
Recibiendo del cliente:
	69
Recibiendo del cliente:
	68
```

Ejemplo de ejecución de un cliente:

```
PROGRAMA CLIENTE INICIADO... Vamos a adivinar un número.
Dime un número para ver si es el que piensa el Servidor (entre 1 y 100):
50
Recibiendo del servidor:
	Número demasiado pequeño
Dime un número para ver si es el que piensa el Servidor (entre 1 y 100):
80
Recibiendo del servidor:
	Número demasiado grande
Dime un número para ver si es el que piensa el Servidor (entre 1 y 100):
60
Recibiendo del servidor:
	Número demasiado pequeño
Dime un número para ver si es el que piensa el Servidor (entre 1 y 100):
65
Recibiendo del servidor:
	Número demasiado pequeño
Dime un número para ver si es el que piensa el Servidor (entre 1 y 100):
70
```

```
Recibiendo del servidor:
       Número demasiado grande
Dime un número para ver si es el que piensa el Servidor (entre 1 y 100):
69
Recibiendo del servidor:
       Número demasiado grande
Dime un número para ver si es el que piensa el Servidor (entre 1 y 100):
68
Recibiendo del servidor:
       CORRECTO
```

EJERCICIOS TIPOS EXAMEN

2.- Examen febrero 2021 - Ejercicio 4. [2 puntos] Servidor de tiempo.

Implementar un programa **servidor TCP** llamado *Ejercicio04.java* que escuche por el **puerto 2021** y que sólo sea capaz atender a un único cliente (servidor no concurrente). Este servidor tan solo tendrá que entender un comando: la palabra "***TIME***". Cuando reciba este comando desde un cliente, el servidor devolverá la **hora local del servidor** en el formato que tú prefieras.

El resultado en pantalla de la ejecución del programa *Ejercicio04* debería ser el siguiente:

```
SERVIDOR DE TIEMPO
------------------
Servidor de Profesor
Servidor iniciado.
Escuchando por el puerto 2021.
Esperando conexión con cliente.
```

En tu caso deberá aparecer tu nombre en lugar de la palabra "*Profesor*". Y así se quedará "esperando" hasta que algún cliente le llegue a solicitar la hora actual a través del comando "*TIME*". Cuando finalmente reciba este comando, se le enviará a la hora local del servidor al cliente que la haya solicitado y el programa servidor finalizará su ejecución ordenadamente. En caso de recibir cualquier otro tipo de comando o petición que no sea "TIME" se deja a opción del alumnado que decida qué hacer (enviar un mensaje de error, finalizar, no hacer nada, etc.).

3.- Examen febrero 2021 - Ejercicio 5. [2 puntos] Cliente de tiempo.

Implementar un programa cliente TCP llamado *Ejercicio05.java* que envíe una solicitud de tiempo (comando "*TIME*") al posible servidor de tiempo que haya escuchando en el puerto 2021 de la propia máquina en la que se ejecute el cliente. El programa debería mostrar por pantalla la información recibida desde el servidor.

Si había algún servidor escuchando en el puerto y se recibe una respuesta, el programa debería mostrar la siguiente información por pantalla:

```
CLIENTE DE TIEMPO
------------------
Cliente de Profesor
Conectándose a localhost por el puerto 2021.

Solicitando información del tiempo local.
Información recibida del servidor de tiempo: TIME: 17:41:38
```

En tu caso deberá aparecer tu nombre en lugar de la palabra "*Profesor*".

Si por el contrario, no había ningún servidor escuchando por el puerto especificado, el resultado debería ser este otro:

```
CLIENTE DE TIEMPO
------------------
Cliente de Profesor
Conectándose a localhost por el puerto 2021.
Error: No se ha podido establecer conexión con el servidor.
```

En ambos casos, el programa finalizaría su ejecución.

4.- Examen febrero 2022 – Ejercicio 4 [2,25 puntos]

Implementar un programa **servidor TCP** llamado *Ejercicio4.java* que escuche por el puerto 6000 y que solo sea capaz de atender a un único cliente (servidor no concurrente). Este servidor recibirá, como cadena de caracteres, un número y, en caso de que se pueda convertir a valor tipo *long* sin incidencias (tratamiento de excepciones) devuelva al cliente si el número es o no es primo. En caso de que no se pueda convertir a valor tipo *long* (por ejemplo, si el cliente envía "Cinco" en lugar de "5", el servidor debería remitir al cliente el mensaje *"El valor introducido no es numérico"*.

El resultado en pantalla de la ejecución del programa *Ejercicio4* debería ser el siguiente:

```
SERVIDOR NO CONCURRENTE DE NÚMEROS PRIMOS.
----------------------------------------
Servidor del alumno XXX iniciado en el puerto 6000
Esperando al cliente...
```

donde, en lugar de "XXX" cada alumno deberá escribir su nombre y primer apellido.

Finalmente, cuando reciba el número, en función de si el número es o no es correcto, deberá dar la respuesta correcta al cliente, mostrando en pantalla lo siguiente:

```
Recibiendo del cliente:
HOLA
Respuesta al cliente: El valor introducido no es numérico
FIN DE EJECUCIÓN DEL SERVIDOR.
```

en caso de recepción de valores incorrectos, o, en caso contrario:

```
Recibiendo del cliente:
2347
Respuesta al cliente: El número 2347 es primo.
FIN DE EJECUCIÓN DEL SERVIDOR.
```

5.- Examen febrero 2022 – Ejercicio 5 [2,25 puntos]

Implementar un programa cliente TCP llamado *Ejercicio5.java* que pida a un usuario un número, lo lea desde entrada estándar (no haciendo ningún tipo de filtro), envíe dicho valor al servidor que se implementó en el ejercicio 4 (para lo cual se tendrá que conectar a dicho servidor mediante los parámetros: dirección=localhost, puerto=6000) y muestre por pantalla el resultado que se debe recepcionar del servidor del ejercicio 4.

En caso de que el servidor del ejercicio 4 esté escuchando en el puerto indicado y envíe correctamente la respuesta, el programa de este ejercicio debería mostrar la siguiente información por pantalla (en función de si el usuario proporciona un valor numérico o no):

Si el usuario escribe un valor no numérico:

```
CLIENTE PARA EVALUAR LA PRIMALIDAD DE NÚMEROS.
--------------------------------------------
Cliente del alumno XXX conectado con localhost e iniciado en el puerto 6000
Dame un número positivo para comprobar su primalidad: HOLA
Recibiendo del servidor:
El valor introducido no es numérico
FIN DE EJECUCIÓN DEL CLIENTE.
```

Y, en caso de que el usuario si escriba el número adecuado:

```
CLIENTE PARA EVALUAR LA PRIMALIDAD DE NÚMEROS.
-----------------------------------------------
Cliente del alumno XXX conectado con localhost e iniciado en el puerto 6000
Dame un número positivo para comprobar su primalidad: 2347
Recibiendo del servidor:
El número 2347 es primo.
FIN DE EJECUCIÓN DEL CLIENTE.
```

donde, en lugar de "XXX" cada alumno deberá escribir su nombre y primer apellido.

Si, por el contrario, no había ningún servidor escuchando por el puerto especificado, el resultado debería ser este otro:

```
CLIENTE PARA EVALUAR LA PRIMALIDAD DE NÚMEROS.
-----------------------------------------------
Error: No se ha podido establecer la conexión con el servidor
FIN DE EJECUCIÓN DEL CLIENTE.
```

En todos los casos, el programa finalizaría su ejecución.

6.- Examen junio 2022 – Ejercicio 2 [4,00 puntos] Hilos y Sockets

Implementar un servidor Socket concurrente y que atienda hasta 100 llamadas de clientes Sockets para el juego del ahorcado.

Para ello, en el proyecto base se facilita el Cliente, que no es concurrente, sino que lanza un único proceso con el que interactúa con el servidor (este cliente puede ser invocado múltiples veces, ya que el servidor es concurrente y tiene un límite muy alto de clientes simultáneos).

Se deberán implementar las clases MainServidor.java y Servidor.java, de tal forma que:

- MainServidor.java cree un nuevo hilo para dar respuesta a todos los clientes que se generen (hasta 100 simultáneamente).
- Servidor.java que conteste las preguntas enviadas por un cliente con la evolución del juego.