Miguel Ángel Ramírez Pérez mramper0604@g.educaand.es

Explicación - Ejercicio 02 - Cliente con Archivo

Introducción

En este ejercicio se pide ampliar el Ejercicio 01, permitiendo que el cliente envíe comando o texto desde un archivo de texto en lugar de enviarlos por la consola.

El sistema funciona de la siguiente forma:

- El servidor es el mismo que el del Ejercicio 01 (puerto **54321**).
- El cliente se conecta al servidor.
- El usuario especifica la ruta del archivo de texto.
- El cliente lee el archivo línea a línea y envía cada línea al servidor.
- El servidor procesa cada comando y responde como en el ejercicio anterior.
- La comunicación finaliza cuando se encuentra el comando ***#fin***.

ClienteArchivo

Gestiona la conexión al servidor y el envío de comandos desde el archivo de texto.

¿Qué es BufferedReader con FileReader y cómo funciona?

BufferedReader y FileReader son clases que permiten leer archivos de texto:

- **FileReader(ruta):** Abre un archivo de texto para la lectura.
- **BufferedReader(FileReader):** Envuelve al FileReader para leer línea a línea.
- **readLine():** Lee una línea completa del archivo (hasta encontrar un salto de línea.)
- **Return null:** Cuando llega al final del archivo.

Ejemplo - Sistemas de pedido automatizados: Una cafetería tiene un sistema donde se preparan los pedidos del día según una lista:

- **Archivo de pedidos:** Contiene la lista de bebidas a preparar
- **FileReader:** Abre un archivo de pedidos.
- **BufferedReader:** Lee pedido a pedido.
- **Envío al servidor:** Cada pedido se transmite a la cocina (servidor)
- **Respuesta:** La cocina confirma cada pedido preparado.
- **#fin en el archivo:** Cuando se lee esta línea, se cierra el turno.

Diferencias con el Cliente del Ejercicio 01

Aspecto	**Ejercicio 01 (Cliente)**	**Ejercicio 02 (ClienteArchivo)**
Origen de datos	Entrada manual por consola (Scanner)	Lectura automática desde un archivo
Método principal	enviarYRecibir()	enviarDesdeArchivo()
Interacción	Usuario escribe desde comando	Usuario solo especifica la ruta del archivo
Bucle de lectura	while(true) con sc.nextLine()	while((linea = fileReader.readLine())

		!= null)
Control de flujo	Usuario decide cuándo enviar cada mensaje	Envío automático línea a línea

Atributos

- **Socket clienteSocket:** Socket para la conexión con el servidor
- **PrintWriter printWriter:** Flujo de salida para enviar mensajes al servidor
- **BufferedReader reader:** Flujo de entrada para recibir respuesta del servidor
- **String HOST:** Dirección del servidor (**"localhost"**).
- **int PUERTO:** Puerto de conexión (**54321**).

Métodos

- **conectar():** Establece la conexión con el servidor. Idéntico al Ejercicio 01. Return **true** si la conexión fue exitosa, **false** en caso contrario.

- **enviardesdeArchivo():** Método principal que gestiona el envío automatizado:
  a. Solicita al usuario la ruta completa del archivo.
  b. Abre el archivo con **FileReader** y **BufferedReader**.
  c. Lee el archivo línea a línea.
  d. Envía cada línea al servidor
  e. Usa **flush()** para asegurar el envío inmediato.
  f. Lee la respuesta del servidor después de cada envío.
  g. Si encuentra **#fin**, lee la respuesta final y termina.
  h. Cierra el archivo y la conexión en el bloque **finally**.

- **leerRespuestaCompleta(boolean comando):** Lee todas las líneas de respuesta del servidor hasta que encuentre el marcador fin. El **comando** indica si es la respuesta al comando **#fin**.

- **cerrar():** Cierra todos los recursos (**PrintWriter**, **BufferedReader**, **Socket**) de forma segura.

Ejecución

1. **Conecta al servidor ->** Crea el socket y los flujos de comunicación.
2. **Solicita la ruta del archivo ->** El usuario introduce la ruta completa del archivo.
3. **Abre el archivo ->** Crea **FileReader** y **BufferedReader** para lectura.
4. **Lee línea a línea ->** Procesa cada línea del archivo secuencialmente.
5. **Envía al servidor ->** Transmite cada línea mediante **PrintWriter** con **flush()**.
6. **Lee la respuesta ->** Recibe y muestra la respuesta del servidor.
7. **Detecta #fin ->** Si encuentra el comando finaliza el bucle.
8. **Cierra los recursos ->** En el **finally** garantiza que tanto el archivo como la conexión se cierran correctamente.
   El **finally** asegura que no queden recursos abiertos (archivo, socket, streams) incluso si ocurre un error durante la lectura o comunicación.

Servidor

El servidor utilizado es el mismo del Ejercicio 01. Procesa los comandos de la misma manera.

- **#fin:** Cierra la conexión.
- **#info:** Envía información del host y puerto.
- **#list:** Lista el contenido del directorio configurado.
- **Mensajes normales:** Confirma recepción.

El servidor no necesita saber si los comandos vienen de entrada manual o de archivo, ya que recibe líneas de texto de la misma forma.

Archivo de comandos

Para utilizar este cliente, se necesita crear un archivo de texto con los comandos/mensajes que se quiera enviar. Por ejemplo:

```
#info

Hola, mi nombre es Miguel Angel

#list

Esto es un ejercicio de PSP

#fin

Prueba despues del fin
```

Formato del archivo

- Cada línea representa un mensaje o comando.
- Las líneas vacías se ignoran automáticamente.
- El comando **#fin** debe ser la última línea para cerrar correctamente la conexión.
- No es necesario agregar el marcador **--- FIN RESPUESTA ---** (lo gestiona el servidor).

Main

Servidor:

```
-> Servidor: Especifica el directorio: C:\Users\Miguel
Angel\IdeaProjects\PSP_2DAM\src\unidad03\ejercicios
-> Servidor: Directorio: C:\Users\Miguel
Angel\IdeaProjects\PSP_2DAM\src\unidad03\ejercicios
-> Servidor: Esperando conexión en el puerto 54321.
-> Servidor: Cliente conectado desde 127.0.0.1
```

ClienteArchivo:

```
-> Cliente: Conectado al servidor (localhost: 54321)
-> Cliente: Introduzca la ruta completa del fichero a enviar:
C:\Users\Miguel
Angel\IdeaProjects\PSP_2DAM\src\unidad03\ejercicios\FicheroConvers
acion
-> Cliente: Iniciando el envío de mensajes desde el archivo
'C:\Users\Miguel
Angel\IdeaProjects\PSP_2DAM\src\unidad03\ejercicios\FicheroConvers
acion'...

-> Cliente (Enviando): #info
--- RESPUESTA DEL SERVIDOR ---
Info: Host: PC-MiguelAngel. Puerto: 54321
------------------------------

-> Cliente (Enviando): Hola, mi nombre es Miguel Angel
--- RESPUESTA DEL SERVIDOR ---
Mensaje recibido correctamente.
------------------------------

-> Cliente (Enviando): #list
--- RESPUESTA DEL SERVIDOR ---
Lista: Contenido de C:\Users\Miguel
Angel\IdeaProjects\PSP_2DAM\src\unidad03\ejercicios:
D ejercicio01
D ejercicio02
F FicheroConversacion
------------------------------

-> Cliente (Enviando): Esto es un ejercicio de PSP
--- RESPUESTA DEL SERVIDOR ---
Mensaje recibido correctamente.
------------------------------

-> Cliente (Enviando): #fin
Fin: Cerrando conexión
-> Cliente: Conexión terminada.
```

Servidor (despues del #fin):

```
-> Servidor (Recibido): #info
-> Servidor (Recibido): Hola, mi nombre es Miguel Angel
-> Servidor (Recibido): #list
-> Servidor (Recibido): Esto es un ejercicio de PSP
-> Servidor (Recibido): #fin
-> Servidor: Conexión cerrada.
```