índice

Enunciado Tarea

Objetivo:

El objetivo de esta práctica es que los alumnos diseñen una pequeña aplicación de escritorio que permita gestionar archivos y manipular su contenido. Los estudiantes pueden elegir entre una interfaz de consola o crear una interfaz gráfica (usando Swing o JavaFX) para implementar las funcionalidades de la aplicación.

Temática propuesta (personalizable):

Cada alumno debe seleccionar una funcionalidad útil y práctica. A continuación se presenta un ejemplo, pero la temática final puede ser elegida por cada estudiante.

Instrucciones:

1. Diseño de la funcionalidad principal

- **Temática:** Los alumnos deben pensar en una funcionalidad práctica, como contar palabras, encontrar números, o realizar operaciones de lectura y escritura en archivos.
- **Clases Java:** Los alumnos deben basar su implementación en algunas de las siguientes clases, vistas en los temas 1 y 2, tales como:
  - **File**, **FileReader**, **FileWriter** para la gestión de archivos de texto.
  - **BufferedReader** y **BufferedWriter** para mejorar la eficiencia en la lectura/escritura de archivos.
  - **StreamTokenizer** para dividir el texto en palabras y números si es necesario.
  - **DataOutputStream** y **DataInputStream** para almacenar y leer datos binarios.

2. Implementación de la aplicación

- **Consola o GUI**: Los alumnos pueden elegir entre una aplicación que funcione en la consola o utilizar una biblioteca de interfaces gráficas para crear una pequeña ventana que interactúe con los archivos.
- **Funciones mínimas**:
  - Leer un archivo de texto y mostrar su contenido.
  - Contar las palabras, números, o realizar alguna operación definida en el archivo.
  - Guardar los resultados (si aplica) en un archivo de salida o mostrar la información al usuario.

3. Interfaz gráfica (opcional)

- Los alumnos que opten por una GUI pueden diseñar una ventana sencilla que permita al usuario:
  - Seleccionar un archivo de texto.
  - Ver el contenido del archivo en un cuadro de texto.
  - Realizar alguna acción en base a la funcionalidad principal (contar palabras, por ejemplo).
  - Mostrar el resultado en la interfaz, como se ve en la imagen que has compartido.

4. Manejo de excepciones

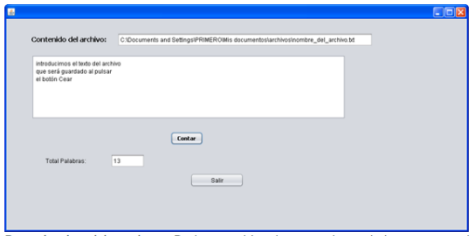
- La aplicación debe manejar posibles errores que puedan surgir, como:
  - Archivo no encontrado.
  - Problemas de lectura o escritura.
  - Archivo vacío o malformado.

5. Documentación del proyecto

- **Explicación del proyecto:** Los alumnos deben escribir un pequeño documento (máximo 1 página) explicando la funcionalidad de la aplicación que diseñaron y las clases de Java que utilizaron.
- **Elección de las clases y métodos:** Deben justificar por qué eligieron las clases mencionadas, cómo implementaron las operaciones de lectura/escritura de archivos, y qué métodos fueron utilizados.
- **Entrega del código fuente.**

Ejemplos de proyectos posibles:

1. **Contador de palabras y caracteres**: La aplicación cuenta cuántas palabras y caracteres hay en un archivo de texto y muestra el resultado.



2. **Buscador de palabras claves**: Dado un archivo de texto, el usuario ingresa una palabra clave, y la aplicación cuenta cuántas veces aparece esa palabra en el archivo.

3. **Conversión de formato**: Leer un archivo de texto y convertir todo su contenido a mayúsculas o minúsculas, guardando el resultado en un nuevo archivo.

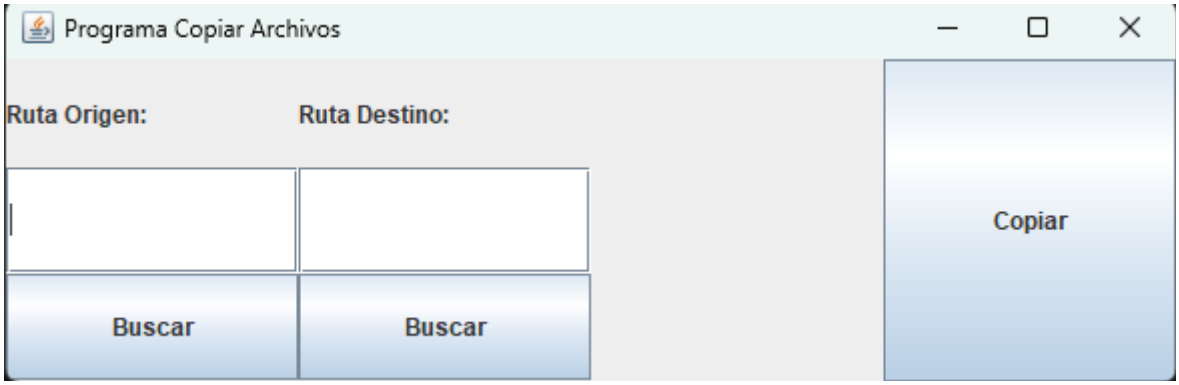
4. **Filtro de números**: Leer un archivo de texto y extraer todos los números que contiene, mostrando los resultados o guardándolos en un archivo separado.

5. **Comunicación entre hilos con tuberías (`PipedInputStream` y `PipedOutputStream`)**:
   - Implementa una aplicación que cree **dos hilos**. Un hilo genera datos (por ejemplo, números aleatorios o texto) y los envía a través de una tubería (piped stream), y el otro hilo recibe esos datos y los muestra por pantalla o los guarda en un archivo.

Copiar y pegar imágenes o archivos binarios

1. Diseño de la funcionalidad principal

- **Temática:** La temática de mi aplicación es para copiar imagenes o archivos binarios con la ruta de accceso donde se encuentra y la ruta de acceso donde queremos pegarla con el nombre nuevo.



Clases Java:

- BufferedInputStream
- BufferedOutputStream
- FileInputStream
- FileOutputStream
- OException

2. Implementación de la aplicación

- **Consola o GUI:** He elegido hacerlo con GUI, una ventana sencilla.
- **Funciones:**
  - Botón tiene una acción donde se crean dos variables: una para la tura de origen y otro para la ruta de destino. Con el botón buscar también podemos añadir las rutas de destino y origen

- Seguido de un if-else. Donde si el archivo de origen y de destino están vacíos cuando se pulse el botón haga un return (devuelva) y no continué con el programa.
- Luego he creado dos tuberias una de entrada y otra de salida.
- Continuamos con un try-catch. (punto 4)
- Por último mostrar la ventana y centrarla

3. Interfaz gráfica (opcional)

- Tiene Interfaz gráfica
  - **Tres Paneles, y un JFrame**
  - **Jlabel:** Ruta de origen
  - **Jlabel:** Ruta de destino
  - **JtextField:** Mostrar error (si falta alfuna ruta o no se ha podido copiar)
  - **Jbutton:** Botón copiar

4. Manejo de excepciones

- Cuando haya un error al copiar un archivo

  Dentro del try: En La tubería input introduzco el archivo de origen y el de output introduzco la de destino. Creo una variable llamada byteLeido; donde se almacenara lo que vaya leyendo de la imagen. Y un While que va a repetirse hasta que ya no quede ninguna línea en el fichero. Un método flush para asegurarnos de que se copie todo.

  Cambiamos el JtextField con un mensaje si se ha realizado con éxito

  y limpiamos las rutas.

  En el catch. Un mensaje de error se muestra en el TextField.