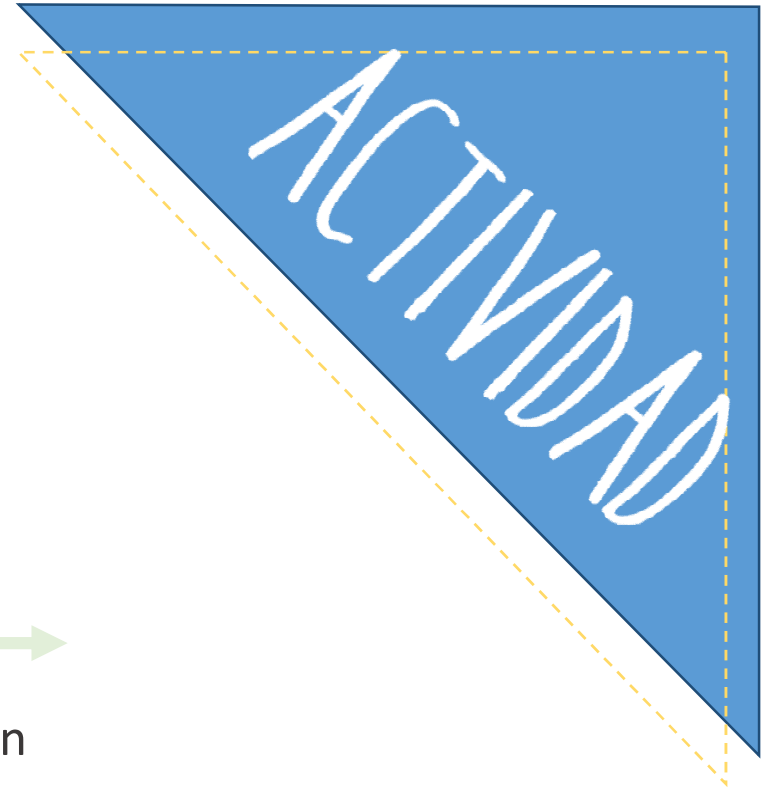

Lectura y escritura de información

Desarrollo de Aplicaciones Web/Desarrollo de Aplicaciones Multiplataforma

Programación

Linkia FP

Actividad

Lectura y escritura de información

Objetivos

- Crear y Realizar operaciones de lectura y escritura sobre ficheros.
- Realizar operaciones de gestión de ficheros y carpetas.
- Modularizar las operaciones sobre ficheros.

¿Cómo lo hago?

1. Rellena los datos que se piden en la tabla“Antes de empezar”.
2. Haz uso de fuentes comunes como Arial, Calibri, Times New Roman etc.
3. Utiliza el color negro para desarrollar tus respuestas y usa otros colores para destacar contenidos o palabras que creas necesario resaltar.
4. Entrega un zip que contenga todos los archivos. java que has creado. Para poder aprobar un ejercicio, éste debe poder ejecutarse sin errores.
5. Recuerda nombrar el archivo zip siguiendo estas indicaciones:
    - Ciclo_Módulo o crédito_Tema_ACT_númeroactividad_Nombre y apellido
        - Ejemplo: AF_M01_T01_ACT_01_Maria Garcia

Antes de empezar…	
Nombre	GARA
Apellidos	GONZÁLEZ SOSA
Módulo/Crédito	M03 PROGRAMACIÓN
UF (solo ciclos LOE)	UF3: FUNDAMENTOS DE GESTIÓN DE FICHEROS
Título de la actividad	LECTURA Y ESCRITURA DE INFORMACIÓN

Se debe entregar un zip que contenga todos los archivos. java que has creado. Para poder aprobar un ejercicio, éste debe poder ejecutarse sin errores. Crea los archivos .java dentro de una carpeta de nombre actividad04

1. **Comenta cada función y clase indicando su objetivo .**Podéis utilizar las clases y funciones creadas anteriormente para pedir valores al usuario. Se valorará el control de errores al introducir datos desde consola.No olvidéis que incluir en el package todas aquellas clases que habéis creado y utilizado para la práctica.

   Main: función principal encargada de generar el menú y llamar a los submétodos.

   Clase Pregunta y función pide entero: le entra por parámetros un String desde el main y la entrada del scanner. En mi programa principal creo una instancia del scanner y esa instancia se la paso a varios submétodos para pedir datos al usuario. En el mismo programa principal la cierro. Le ponemos la función do/while y try/catch para tratar posibles errores.

2. Para indicar la ruta de las carpetas y archivos debes utilizar las funciones: System.getProperty("user.dir"); y File.separator;
3. **Ejercicio01.java**: muestra por consola un menú con las siguientes opciones (y programa cada opción en una función aparte) teniendo presente que si no existe una carpeta de nombre "archivos" ubicada en la carpeta del proyecto, se cree de manera que quedará en la misma carpeta que las carpetas src y bin:

Las opciones del menú serán:

3.1. **Nuevo Archivo**: debe llamar a una función que pida un nombre de archivo al usuario y cree un archivo con el nombre indicado dentro de la carpeta de nombre "archivos" ubicada en la carpeta del proyecto, y que rellene el archivo con un texto introducido por el usuario.

crearArchivo: Pide al usuario que indique el nombre del archivo y el texto a introducir. Le entra por parámetros 2 Strings con la ruta de la carpeta y el separador los cuales necesitamos para conocer la ruta en donde se va a crear el archivo.
Funciones utilizadas en crearArchivo:
La función: newFile.createNewFile(); crea el archivo en la ubicación ya establecida a travez de la ruta indicada.
Creamos un Stream de escritura vinculado con el archivo en el queremos escribir fw= new FileWriter(ruta del archivo).
BufferedWriter bw= new BufferedWriter(fw) : Creamos el buffer de escritura y lo vinculamos con el Stream.
Bw.write : escribe el texto el el archivo.
bw.flush(); //obliga al buffer a escribir en el archivo todo quello que contenga el buffer. (lo vacia)
bw.close(); //vacia el buffer y libera el archivo cerrando la conexion con él.

3.2. **Listar Archivos** debe llamar a una función que muestre los nombres de archivos (sin carpetas) dentro de la carpeta "archivos" numerados: ej: 1-Archivo1.txt 2-Archivp2.txt .. y retorne un array con las rutas de los archivos.

listarArchivos: Creamos una función de tipo Array para listar los archivos.
Le entra por parámetros String con la ruta de la carpeta y el separador los cuales necesitamos para conocer la ruta en donde van a estar los archivos.
Este método muestra por consola el listado de los archivos númerados. Devuelve el Array con el listado de archivos.

carpeta.list(); // crea un array de Strings con los nombres de los archivos y carpetas dentro del carpeta definida en la variable File.

3.3. **Muestra un Archivo** debe de llamar a **ListarArchivos** para mostrar los archivos disponibles y permitir al usuario elegir qué documento quiere ver según su número y mostrar el contenido del documento por consola.

muestraArchivo: Creamos una función que pida el usuario la posición del archivo que desea ver y se lo muestra. Le entra por parámetros String con la ruta de la carpeta y el separador los cuales necesitamos para conocer la ruta en donde va a estar el archivo seleccionado.

Este método muestra por consola el contenido del archivo.
Inicializamos la i=-1, para solventar el problema de que no pase un caracter no numerico si el usuario escribe letras.
En la condición del while contralamos que el valor introducido por el usuario sea el tamaño del array.

3.4. **Borrar un Archivo**: debe llamar a una función que muestre los archivos dentro de la carpeta "archivos" numerados y permitir al usuario elegir qué documento quiere borrar según su número.

borrarArchivo: creamos una función que pida el usuario la posición del archivo que desea borrar y lo borra. Le entra por parámetros String con la ruta de la carpeta y el separador los cuales necesitamos para conocer la ruta en donde va a estar el archivo seleccionado.
Este método muestra borra el archivo seleccionado por el usuario.
archivo.delete(); función que borrar el archivo seleccionado. Anteriormente tuvimos que indicar la ruta del archivo.

3.5. **Renombrar un Archivo**: debe mostrar los archivos dentro de la carpeta "archivos" numerados y permitir al usuario elegir qué documento quiere renombrar según su número. A continuación, le pregunte el nuevo nombre y lo renombre si es válido. Si es un nombre inválido se debe mostrar un mensaje por consola al usuario y volver a ejecutar el menú.

renombrarArchivo: Creamos una función que pida el usuario la posición del archivo al que le desea cambiar el nombre. Le entra por parámetros String con la ruta de la carpeta y el separador los cuales necesitamos para conocer la ruta en donde va a estar el archivo seleccionado + la entrada del scanner Este método modifica el nombre del archivo y lo muestra por consola.
archivo.renameTo(newName); función que renombra el archivo.
Estructuras de control:
Con un do/while controlamos que pide entero sea un número válido (del tamaño del array)
Con el for y un If aplicamos la condición de si el nuevo nombre ya existe en nuestra carpeta no nos lo deje guardar.

3.6. **Reemplazar caracteres de un Archivo utilizando RandomAccessFile**: debe de llamar a **ListarArchivos** para permitir al usuario elegir qué documento se quiere modificar según su número y a continuación pida qué carácter se quiere reemplazar y por qué nuevo carácter. Si el documento no existe o es inválido, se debe mostrar un mensaje por consola al usuario y volver a ejecutar el menú. Es obligatorio utilizar el **RandomAccessFile** para éste punto.

reemplazarCaracteres: Creamos una función que reemplace un caracter por otro que elija el usuario.Parámetros de entrada String con la ruta de la carpeta y el separador los cuales necesitamos para conocer la ruta en donde va a estar el archivo seleccionado + la entrada del scanner. Este método modifica el caracter por otro que elija el usuario.
Con un do/while controlamos que pide entero sea un número válido (del tamaño del array).
RandomAccessFile raf = new RandomAccessFile- >raf: variable del tipo RandomAccessFile por defecto tiene asociado un puntero que apunta a una posición de memoria.
rw variable modo acceso al archivo lectura y escritura imprimim el texto del fichero
(raf.getFilePointer() < raf.length()) { //Retorna la posicion actual del puntero, inicialmente mientras el puntero no llegue al final del archivo (texto dentro del fichero), se va a ir ejecutando
Character chr = (char)raf.readByte(); // readbyte: cada caractaer que va encontrando lo va almacenando + Aumenta en 1 el puntero.
if(chr.toString().compareTo(text_find)==0){ // si el caracter que coge es el que hemos dicho de modificar entonces hace lo siguiente:
raf.seek(raf.getFilePointer()-1); //si lo que encontraste era el caracter que estaba buscando aqui vas para atras otra vez-mover el puntero a la posicion 0 del archivo
raf.writeBytes(text_replace); //a partir del puntero se se sustituyen los bytes siguientes por cada carácter del String pasado como parámetro. Reemplaza el caracter
raf.close(); //cerramos fichero.

3.7. **Salir:** debe terminar el programa. Si el usuario selecciona una opción no contemplada se tiene que volver a mostrar el menú.