ESTRUCTURAS DE DATOS Y ALGORITMOS II **2021-1**

PRÁCTICA #10: ARCHIVOS

OBJETIVO: El ESTUDIANTE CONOCERÁ E IDENTIFICARÁ ASPECTOS SOBRE LOS ARCHIVOS, COMO LAS OPERACIONES, EL TIPO DE ACCESO Y ORGANIZACIÓN LÓGICA

ACTIVIDADES PARA EL DESARROLLO DE LA PRÁCTICA

Integrantes:

Díaz Hernández Marcos Bryan

Lara Aguilar Christian Abraham

I.- Responde las siguientes preguntas relacionadas con implementación de archivos en Java

1.- Investiga en la api de Java para qué sirve la clase File, y para qué sirven los siguientes métodos

La clase File sirve para poder trabajar con archivos, es decir llevar a cabo distintos métodos como la creación, la eliminación, la escritura, la lectura de documentos en Java

- canRead(): verifica que se pueda leer el archivo que se indica en la ruta que indica al archivo, en caso de que no se pueda leer regresa un false, y en caso de que si se pueda leer regresa un true.
- canWrite(): verifica que sea posible el escribir datos en un archivo que se indica en la ruta, puede darse el caso de que solo se permite la lectura pero no la modificación, en caso de que si permite editar se regresa un true, en caso de que no se regresa un false.
- createNewFile(): crea un nuevo archivo vacío, en la ruta y con el nombre que se le de en el path de archivo, únicamente en caso de que el archivo no exista, regresa un true en caso de que se haya creado, y un false en caso de que exista.
- getName(): retorna el nombre de la carpeta en donde se encuentra el archivo.
- getPath():retorna la dirección completa del archivo, indicando toda la ruta que tiene el archivo dentro del sistema en forma de una cadena.
- length(): regresa el tamaño del archivo, que indica las líneas de texto que lo componen.

2.- Investiga las principales diferencias entre las clases

- File Writer y BufferedWriter

  La clase BufferedWriter hace uso de un búfer, el cual almacena la secuencia de caracteres que se quiera escribir en el búfer y ya después los escribe en el archivo. Por lo tanto hace una sóla llamada con la cadena de caracteres almacenados en búfer, lo que lo hace eficiente si no se sobrepasa del tamaño del búfer pre-establecido (igualmente se puede modificar).

  La clase FileWriter igualmente sirve para escribir, principalmente en archivos, pero lo hace de una forma directa, sin tener que recurrir a un búfer.

- FileReader y BufferedReader

Al igual que en el caso anterior, BufferedReader hace uso de un búfer en el cual se almacena, mediante un stream, la información que queramos leer. E igualmente, FileReader lo va haciendo por vez. FileReader funciona mejor al leer bytes ya que lo hace de poco en poco, a diferencia de BufferedReader.

Ambas clases se pueden juntar para mejorar la eficiencia al leer (o al escribir, en el caso anterior).

3.- ¿Cómo hace el programa proporcionado para establecer la diferencia entre agregar contenido a un archivo o sobreescribir la información anterior?

Con un booleano en el método FileWriter, que indica si la información que se quiere agregar debe de concatenarse (append) a la información que se tenía previamente, o si se debe sobreescribir la información del archivo existente.

4.- ¿Qué ocurre si el usuario desea agregar contenido a un archivo pero no indica la extensión? ¿Por qué?

Si se quiere agregar, es decir, utilizar un Writer pero no se especifica la extensión, se creará el archivo pero no podrá ser abierto de forma normal, ya que al no tener una extensión, el sistema operativo no reconocería con qué programas pueden ser abiertos. Igualmente se podría abrir si se cambia la extensión.

5.- ¿Por qué el programa proporcionado no maneja la excepción (no tiene un try-catch) FileNotFoundException en ningún lugar, y esa excepción no ocurre cuando el usuario ingresa un archivo que no existe?

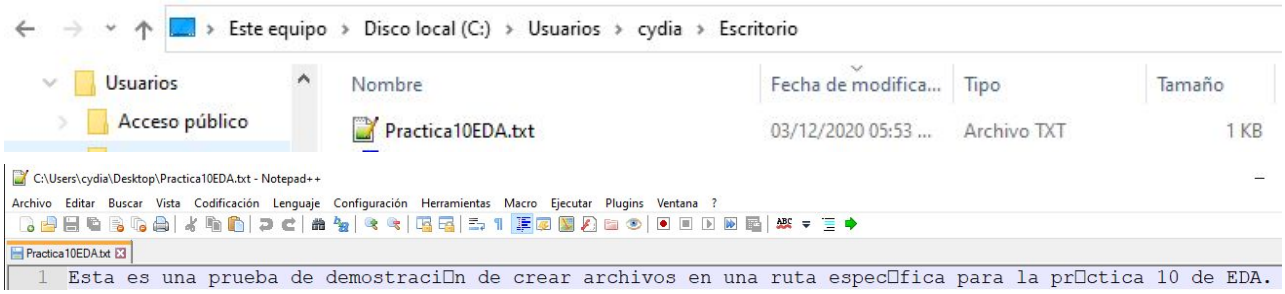
Porque se indica, mediante un if, que la operación sólo se ejecutará si el archivo existe, si no, ni siquiera se abre el archivo mediante FileWriter o se ejecuta el método para eliminarlo, por lo que la excepción no puede ser lanzada si no se ejecuta el método que la lanza.

6.- Modifica el programa de tal manera que, al momento de crear un nuevo archivo, el usuario indique la ruta donde el lo quiera crear (por ejemplo el escritorio)
Agrega en este cuestionario un recorte con el fragmento o fragmentos de código modificados y una imagen del programa corriendo donde se vea claramente que el archivo si se crea en la ruta ingresada por el usuario

```
1.Crear Archivo
2.Sobreescribir en el archivo
3.Añadir contenido en el archivo
4.Elimina el archivo
5.Salir

1
Ingresa la ruta completa y nombre del archivo donde quieras que se almacene (sin extensión).
Ejemplo:
C:\Users\TU_USUARIO\Desktop\NOMBRE_ARCHIVO
C:\Users\cydia\Desktop\Practica10EDA
Se creó el archivo exitosamente.

Añadir contenido:
Esta es una prueba de demostración de crear archivos en una ruta específica para la práctica 10 de EDA.
```



7.- Indica qué tendrías que agregar al programa para que el usuario pueda eliminar una carpeta con archivos

Para poder eliminar la carpeta que contiene archivos se tiene que crear un método de eliminación que reciba la dirección de la carpeta que se quiere eliminar, y se crea un objeto del tipo File en la dirección a eliminar, posteriormente se crea un arreglo de archivos que contendrá todos los nombres de los archivos que estén presentes en el directorio de la carpeta mediante el objeto FIle que se creó, posteriormente se tendrá que pasar a ciclo for que obtendrá el número de nombres dentro del archivo y con una referencia de File que recibe las direcciones comienza a eliminar archivo a archivo hasta que se eliminen todos los archivos.

```
public void eliminarCarpeta(String direccion) {
    File carpeta = new File(direccion);
    File[] archivos = carpeta.listFiles();
    File temp;
    if (archivos.length != 0) {
        for (int i = 0; i < archivos.length; i++) {
            temp = new File(archivos[i].toString());
            temp.delete();
        }
    }
```

8.- Utiliza el programa para crear un archivo llamado “con” agrega al reporte evidencia clara de

la creación de ese archivo

La palabra “con”, entre otras, son palabras reservadas para tareas del sistema operativo. Estos nombres son prefijos de palabras, en este caso para “console” el cual está reservado para dispositivos de entrada salida, tal como el teclado y la pantalla. Windows utiliza estas abreviaciones para crear de cuando en cuando archivos que ayudan con la impresión en pantalla.

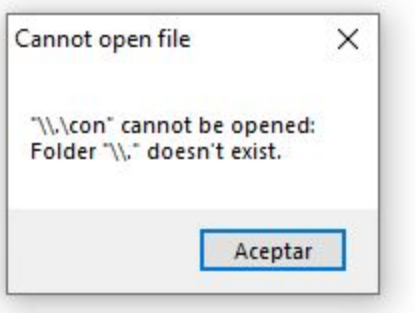
También existen otras palabras reservadas, tales como “CON, PRN, AUX, NUL, COM1, COM2, ..., COM9, LPT1, LPT2, ... , LPT8”, entre otras.

Para crear estos archivos con dichos nombres, se puede hacer agregando “\\.” antes de la ruta donde se desee crear el archivo. Ejemplo:

```
\\.\C:\Users\cydia\Desktop\con
Se creó el archivo exitosamente.

Añadir contenido:
si se pudo :)
```

El gran inconveniente de hacerlo de esta forma, es que siempre se tendría que referir a este archivo mediante su ruta absoluta y no se podría trabajar de una forma cómoda (abriéndolo con doble click, por ejemplo):



En este caso se forza al sistema operativo a crear la carpeta debido a que la palabra está reservada dentro del namespace de los archivos, por lo que mediante “\\.\” es posible acceder a un *namespace* para dispositivos y no para archivos de Windows, lo que permite la creación del archivo.

En otros sistemas operativos si se podría, ya que las palabras reservadas son exclusivas en Windows.

II.- Elabora tus Conclusiones de la Práctica

Debido a que ya se tenía previo conocimiento del manejo de archivos en Java para la creación del proyecto 1 de la materia, nos fue posible realizar la práctica de manera rápida y eficiente.
Los JavaDocs de Oracle nos fueron de mucha ayuda para descubrir cómo trabajan a detalle las clases mencionadas en la práctica, al igual que sus métodos.
Para poder realizar el último ejercicio, tuvimos que realizar una investigación específica acerca de por qué no se pueden crear los directorios con estos nombres reservados, lo que nos llevó a investigar más acerca de las palabras reservadas, las cuales vienen desde MS-DOS y sistemas operativos Windows antiguos, pero que aún se siguen utilizando para diversos usos como en los archivos *batch*.
Al utilizar los directorios aprendimos a distinguir la forma de la creación de los archivos y cómo se trabaja en esto por medio de la ruta relativa y la ruta concreta, que se maneja en la organización del sistema.
Por todo lo anterior y mediante el análisis podemos concluir en que se cumplio el objetivo

Referencias:

- https://docs.microsoft.com/en-us/windows/win32/fileio/naming-a-file