	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorios de docencia

Laboratorio de Computación Salas A y B

Profesor(a): Edgar Tista García

Asignatura: Estructura de Datos y Algoritmos II

Grupo: 3

No de Práctica(s): 10

Integrante(s): Román Ramos María Fernanda

Lopez Gomez Mitzy Valeria

No. de lista o brigada:

Semestre: 2026-1

Fecha de entrega: 07/11/2025

Observaciones:

CALIFICACIÓN: _____

ESTRUCTURAS DE DATOS Y ALGORITMOS II
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

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

1.- ¿Cuál es la diferencia entre la organización lógica y física?

La organización lógica se basa en cómo los archivos se estructuran en la máquina, a partir de carpetas y estas se presentan al usuario, es la interacción con el sistema operativo mientras que la organización física es como se organizan los archivos a nivel hardware, la interacción de los archivos con el disco o la memoria, así como las rutas en las que se almacenan.

2.- Explica cuál es la razón por la cual los archivos se organizan por bloques

Los archivos se organizan por bloques porque se requiere que las operaciones sobre el disco sean más rápidas.

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

- ***canRead()*** *: Lo que hace es devolver un booleano para saber si el archivo se puede o no leer.*
- ***canWrite()*** *: Devuelve un booleano para saber si el archivo se puede escribir o no.*
- ***createNewFile()*** *: Crea un archivo y devuelve un booleano (false) solo si no existía un archivo ya con ese nombre, en caso de que si exista devuelve true.*
- ***getName()*** *: Devuelve el nombre del archivo o directorio*
- ***getPath()*** *: Convierte la ruta del archivo a una ruta de tipo String, por lo que devuelve un String.*
- ***length()*** *: Devuelve la longitud de la ruta de archivo.*

4.- Investiga las principales diferencias entre las clases ¿Por qué se separa la lógica en clases tipo

"Reader" y clases tipo "Writer"

Estas clases resultan ser más prácticas para las aplicaciones en las que se manipula texto.

La clase Reader es para leer texto, lo que son datos de entrada.

La clase Writer es para escribir texto lo que es dato de salida.

La lógica se separa porque una se usa para los datos de entrada y la otra para los datos de salida, es decir sus operaciones son distintas, por lo que la abstracción y modularidad es importante.

FileWriter y BufferedWriter

FileWriter permite escribir archivos de texto directamente manejando cadenas de caracteres sin preocuparnos por la conversión a bytes. Mientras que BufferedWriter añade un buffer para optimizar la escritura, este se crea a partir de FileWriter. Esto es muy útil porque permite trabajar con flujos de datos grandes, debido a que se reduce el número de accesos directos.Buffer Writer se crea a partir de FileWriter.

FileReader y BufferedReader

Cuando se crea un FileReader, le indicamos el archivo que queremos leer y para facilitar la lectura secuencial de cadenas de caracteres, especialmente cuando queremos procesar archivos línea a línea, es mucho más común y eficiente utilizar la clase BufferedReader.

Esta clase actúa como una capa bufferizada sobre un Reader, lo que mejora el rendimiento y nos proporciona métodos muy útiles para trabajar con texto.

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

El programa utiliza el segundo parámetro del constructor de FileWriter. En el método escribir de la clase Archivo, hay un parámetro booleano llamado ***modo****. Cuando se llama a new FileWriter(archivo, modo), si* ***modo*** *es true, el contenido se agregará al final del archivo. si* ***modo*** *es false, el archivo se sobrescribirá.*

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

Si el usuario no indica la extensión del archivo, el programa creará o buscará un archivo con el nombre, sin extensión, no imprime errores

¿Por qué?

porque se trata el nombre completo del archivo, incluida su extensión, como un identificador único, no trata a la extensión como un atributo a parte, u obligatorio, simplemente con el nombre ingresado por el usuario, y FileWtriter solo se encarga de crear el archivo, no necesita una extensión para funcionar, trata el nombre simplemente como una ruta o identificador único. Si la cadena nombre no tiene puntos, java creará un archivo sin extensión, si si tiene, creará uno con esa extensión

7.- ¿Por qué el programa proporcionado no maneja la excepción (no tiene un try-catch)

FileNotFoundException en ningún lugar, y por qué la excepción no ocurre cuando el usuario

ingresa un archivo que no existe?

porque en Archivo evita que ocurra al intentar escribir en los casos de sobrescribir y añadir, el código primero verifica que exista del archivo con archivo.exists(), si el archivo no existe, simplemente muestra un mensaje y no intenta abrir un FileWriter

8.- Indica qué tendrías qué agregar al programa para que el usuario pueda eliminar un directorio con archivos y otros directorios (copiar aquí el código)

Se tendría que recibir el nombre del directorio a eliminar, crear un objeto File con su nombre

1. *verificar que exista con .exists*
2. *si existe, verificar si es un directorio con .isDirectory()*
3. *si es un directorio, crear un arreglo de los archivos que contiene con* ***.listFiles()***
    a. ***el método listFiles() devuelve un arreglo de objetos File, usamos este arreglo porque no existe un método para saber si el directorio está vacío, entonces es la única forma que encontramos para saber si el directorio tiene contenido***
4. *si el arreglo tiene archivos (que el directorio no está vacío), con un for-each, para cada archivo,* ***verificar si es un directorio****, si lo es, llamar recursivamente al método para que haga lo mismo en ese directorio con archivo.getPath(), que devuelve la ruta del archivo o directorio como String, si no es un directorio, eliminar el archivo actual*
    a. *Al llamar recursivamente a la función, permite que se puedan eliminar un directorio con directorios dentro*
5. *eliminar el directorio con delete()*
    a. *delete() regresa un true si se pudo eliminar, y un false si no, entonces se verifica ese boolean para enviar un mensaje si es que no se pudo eliminar*

Código:

```
public void eliminarDirectorio(String nombredir) {
   File dir = new File(nombredir);
   if (dir.exists()) {
      if (dir.isDirectory()) {
         File[] archivos = dir.listFiles();
         if (archivos != null) {
            for (File archivo : archivos) {
               if (archivo.isDirectory()) {
                  eliminarDirectorio(archivo.getPath());
               } else {
                  archivo.delete();
               }
            }
         }
      }
      if (dir.delete()) {
         System.out.println("se eliminó el directorio");
      } else {
```

```
            System.out.println("no se pudo eliminar el directorio");
        }
    } else {
        System.out.println("no existe el directorio");
    }
}
```

II.- Elabora tus Conclusiones de la Práctica

*Conclusiones individuales, respuestas de las preguntas, en equipo

Lopez Gomez Mitzy Valeria

En esta práctica se logró entender cómo es que los archivos se manejan y manipulan, sabemos que sus operaciones básicas son editar, escribir, crear y muchas más como se vio en la clase de teoría, pues con esta práctica se logró comprender cómo estas operaciones funcionan a partir de las clases que java proporciona como File, FileReader, FileWriter, BufferedReader y BufferedWriter. También en la clase de teoría se observó cómo se organizan estas clases a partir de la herencia, como algunas clases heredan de FileWriter y FileReader, y cómo son esenciales para la modificación de texto. Es interesante darse cuenta que todo eso ocurre cuando modificamos un archivo sin darnos cuenta, como eso está sucediendo justo ahora al escribir en este archivo. Es por esto que considero que los objetivos se cumplieron. Gracias a la práctica, a la investigación que se realizó y la clase de teoría fue más fácil comprender cómo las clases de archivos se organizan y se complementan entre ellas.

Como ya se mencionó la clase de teoría sirvió mucho para comprender la práctica.

Román Ramos María Fernanda

La práctica fue útil para conocer muchas funciones de archivos con la clase File y FileWriter de Java, se cumplieron los objetivos ya que pude comprender las operaciones con distintos tipos de archivos (como los archivos normales y directorios) como crear, escribir, sobrescribir y eliminar, fue bastante útil ya que al ir haciendo pruebas con cada opción del menú quedaba más claro su comportamiento para cada caso, y para detectar errores en la parte de eliminarDirectorio, igualmente en este ejercicio se comprendió la parte de la jerarquía de los archivos, ya que pudimos observar cómo se debían eliminar los archivos dentro de él. También se reforzó el concepto sobre la organización de estas clases mediante herencia y su importancia en la modificación de texto.

Referencias

- Class File. Oracle Documents. **https://docs.oracle.com/javase/8/docs/api/java/io/File.html**

- ESTRUCTURA DE DATOS CON ORIENTACIÓN A OBJETO, (s/f). https://unamer34.wordpress.com/wp-content/uploads/2008/04/edypoo.pdf

- Java Files (s.f) https://www.w3schools.com/java/java_files.asp

- File isDirectory() method (2025) File isDirectory() method in Java with Examples - GeeksforGeeks

- Reader, FileReader y BufferedReader, (2022). Makigas. https://www.makigas.es/series/java-io/reader-filereader-y-bufferedreader