TEMARIO: ACCESO A DATOS - MANEJO BÁSICO DE FICHEROS EN JAVA

1. Introducción al manejo de ficheros

1.1 ¿Qué es el acceso a datos?

Es la capacidad de un programa para leer, escribir y manipular información almacenada en el sistema de archivos. En Java, esto incluye trabajar con archivos y directorios.

1.2 Conceptos fundamentales

- **Archivo (File):** Conjunto de datos con un nombre específico.
- **Directorio (Directory/Folder):** Carpeta que contiene archivos y/o otros directorios.
- **Ruta (Path):** Ubicación de un archivo o directorio.
- **Ruta absoluta:** Completa desde la raíz (ej: `C:\Users\Usuario\Documents\archivo.txt`).
- **Ruta relativa:** Relativa a la carpeta actual.

---

2. La clase File en Java

2.1 ¿Qué es la clase File?

Permite trabajar con archivos y directorios, pero **no** leer ni escribir su contenido. Con ella se puede:

- Verificar existencia.
- Crear archivos y directorios.
- Obtener información.
- Navegar por el sistema.

2.2 Importación

```
import java.io.File;
```

2.3 Creación de objetos File

```
File archivo = new File("C:\\ruta\\archivo.txt");

File directorioPadre = new File("C:\\Users\\Usuario\\Documents");
File archivo = new File(directorioPadre, "mi_archivo.txt");
```

El segundo constructor es más flexible y legible.

3. Verificación de archivos y directorios

- **exists():** Verifica si existe.
- **isFile():** Comprueba si es un archivo.
- **isDirectory():** Comprueba si es un directorio.
- **getAbsolutePath():** Devuelve la ruta completa.

---

4. Trabajando con URIs

- **Formato estándar:** `file:///C:/Users/Usuario/Documents/archivo.txt`
- **Importaciones:**

```
import java.net.URI;
import java.net.URISyntaxException;
```

- **De URI a File:**

```
URI uri = new URI("file:///C:/ruta/archivo.txt");
File archivo = new File(uri);
```

- **De File a URI:**

```
File archivo = new File("C:\\archivo.txt");
URI uri = archivo.toURI();
```

---

5. Exploración de directorios

- **list():** Devuelve array con nombres de archivos y carpetas.
- Verificar siempre: `exists()`, `isDirectory()`, `contenido != null`.
- Ejemplo de exploración con tipo de elemento:

```
for (String nombre : directorio.list()) {
    File elemento = new File(directorio, nombre);
    if (elemento.isFile()) System.out.println(nombre + " [ARCHIVO]");
    else if (elemento.isDirectory()) System.out.println(nombre + "
[DIRECTORIO]");
}
```

---

6. Creación de directorios

- **mkdir():** Crea solo si existe el padre.
- **mkdirs():** Crea junto con todos los padres necesarios.

7. Creación de archivos

- **createNewFile():** Crea un archivo vacío.
- Requiere **try-catch** porque puede lanzar `IOException`.
- Antes de crear, se pueden generar los directorios padres con `mkdirs()`.

---

8. Métodos adicionales de File

- `getName()`, `length()`, `getParent()`, `canRead()`, `canWrite()`, `getParentFile()`.

---

9. Buenas prácticas

- Verificar siempre antes de usar métodos.
- Usar constantes para rutas base.
- Manejar errores con `try-catch`.
- Emplear `File.separator` para compatibilidad multiplataforma.

---

10. Integración con Scanner

- Lectura de rutas por teclado.
- Menús con `switch`.
- Confirmaciones (`s/n`).

```
Scanner scanner = new Scanner(System.in);
```

```
String texto = scanner.nextLine();
```

---

11. Organización en funciones

- **Ventajas:** reutilización, organización, mantenimiento.
- Funciones sin parámetros, con parámetros y con valores de retorno.

---

12. Casos prácticos

- Organizador de archivos.
- Explorador de sistema.
- Verificador de integridad.

- Asistente de archivos interactivo.

13. Errores comunes

- No verificar `exists()`.
- Olvidar `try-catch` en `createNewFile()`.
- No limpiar buffer de `Scanner`.
- Uso incorrecto de barras en rutas.

EJERCICIOS DE ACCESO A DATOS - FICHEROS EN JAVA

Ejercicio 7: Organizador de biblioteca

- Función `organizarBiblioteca()`: crea carpeta de categoría y archivo `catalogo.txt`.
- Función `verificarLibro()`: verifica si existe un libro; si no, pregunta si se crea.
- Practica: `exists()`, `mkdir()`, `createNewFile()`, funciones separadas.

Ejercicio 8: Explorador de carpetas inteligente

- Función `explorarCarpeta(String ruta)`: lista contenido.
- Función `analizarElemento(String ruta)`: muestra si es archivo (con tamaño) o carpeta (con número de elementos).
- Función `convertirAURI(String ruta)`: convierte ruta a URI.
- Practica: `list()`, `isFile()`, `isDirectory()`, `toURI()`.

Ejercicio 9: Asistente personal de archivos

Menú principal con opciones:

1. Verificar archivo
2. Explorar carpeta
3. Crear carpeta
4. Crear archivo
5. Trabajar con URIs
6. Salir

Funciones principales: `mostrarMenu()`, `verificarArchivo()`, `explorarDirectorio()`, `crearCarpeta()`, `crearArchivo()`, `trabajarConURI()`.

Requisitos: cada función independiente, uso de `Scanner`, `switch`, `do-while`, manejo de errores.

Criterios de evaluación

- Uso correcto de funciones, parámetros y reutilización de conceptos.
- Código limpio y comentado.
- Menú funcional y manejo de errores.
- Feedback claro al usuario.