

ACTIVIDAD 1.8 - Tratamiento de ficheros CVS, DAO, properties, java.nio

El objetivo de la actividad es el tratamiento de ficheros de texto, así como la información que hay en ellos. En esta actividad se "jugará" con un fichero en formato CSV tanto para la consulta como para la actualización utilizando las clases del paquete java.nio.

▼ Investigación sobre las clases del paquete java.nio

▼ ¿Cuáles son las principales diferencias con las clases del paquete java.io para el tratamiento de ficheros?

`java.io` es el paquete clásico para ficheros. `java.nio` hace lo mismo pero de forma más cómoda y con más posibilidades. Lo más importante:

	`java.io`	`java.nio` (NIO.2)
Ruta	`File` (ruta + operaciones mezcladas)	`Path` (solo la ruta)
Operaciones	Métodos de `File` y streams	Clase `Files` (métodos estáticos)
Errores	Devuelve `true` / `false` sin explicar el fallo	Lanza excepciones claras (`NoSuchFileException` , `FileAlreadyExistsException` ...)
Leer/escribir	Encadenar `FileReader` , `BufferedReader` ...	Métodos directos (`readAllLines` , `write`)
Charset	Opcional, a veces el del sistema	Se indica explícitamente, por defecto UTF-8
Mover/copiar	`renameTo()` limitado	`Files.move()` / `Files.copy()` con opciones

	`java.io`	`java.nio` (NIO.2)
Extras	Pocos	Atributos, enlaces simbólicos, `WatchService`, canales y buffers

▼ Clases y métodos por operación

Las clases que intervienen son `Path`, `Paths`, `Files`, `StandardOpenOption`, `StandarCopyOption`, `Charset` y `StandardCharsets`. Lo primero es crear la ruta:

```
Path ruta = Paths.get("datos", "personas.csv");   // o Path.of("dat
os/personas.csv")
```

Crear, borrar, renombrar y mover

Crear un fichero se hace con `createFile`, y para directorios hay dos opciones: `createDirectory` (un solo nivel, falla si el padre no existe) y `createDirectories` (crea también los padres que falten). Para borrar, `delete` da error si no existe y `deleteIfExists` no. Un directorio solo se puede borrar si está vacío.

En NIO no existe "rename": renombrar es mover dentro de la misma carpeta, así que ambas cosas se hacen con `Files.move`.

Operación	Método
Crear fichero	`Files.createFile(ruta)`
Crear directorio	`Files.createDirectory(dir)` / `Files.createDirectories(dir)`
Borrar	`Files.delete(ruta)` / `Files.deleteIfExists(ruta)`
Renombrar	`Files.move(ruta, ruta.resolveSibling("nuevo.csv"))`
Mover	`Files.move(origen, destino, StandardCopyOption.REPLACE_EXISTING)`
Copiar (extra)	`Files.copy(origen, destino, opciones)`

Abrir un fichero o directorio

Depende de para qué se quiera abrir:

Para…	Método
Leer bytes	`Files.newInputStream(ruta)`
Escribir bytes	`Files.newOutputStream(ruta)`
Leer texto	`Files.newBufferedReader(ruta, charset)`
Escribir texto	`Files.newBufferedWriter(ruta, charset, opciones)`
Acceso aleatorio	`Files.newByteChannel(ruta, opciones)`

Para...	Método
Directorio	`Files.newDirectoryStream(dir)`, `Files.list(dir)`, `Files.walk(dir)`

Al abrir se pueden indicar opciones con `StandardOpenOption`: `READ`, `WRITE`, `APPEND` (añadir al final), `CREATE` (crear si no existe), `CREATE_NEW` (crear y fallar si ya existe) y `TRUNCATE_EXISTING` (vaciar al abrir).

Juego de caracteres (Charset)

El `Charset` indica cómo se convierten los bytes del fichero en letras. Si se elige mal, las tildes y las ñ salen como símbolos raros. Se pueden obtener por nombre o con la clase `StandardCharsets`, que trae los más comunes (`UTF_8`, `UTF_16`, `ISO_8859_1`, `US_ASCII`...), y se pasa como parámetro al leer o escribir:

```
Charset iso  = Charset.forName("ISO-8859-1");
Charset utf8 = StandardCharsets.UTF_8;

Files.newBufferedReader(ruta, iso);
Files.newBufferedWriter(ruta, utf8);
Files.readAllLines(ruta, iso);
```

Leer, escribir y añadir

Para leer todas las líneas como lista de Strings se usa `Files.readAllLines`; cada elemento de la lista es una línea. Si el fichero es muy grande, `Files.lines` devuelve un `Stream<String>` que se va leyendo poco a poco.

Para escribir se usa `Files.write` (con una lista de Strings) o `Files.writeString` (un solo String, Java 11+). **Ojo:** por defecto sobrescriben el fichero si ya existe. Para añadir al final sin borrar lo anterior hay que usar la opción `APPEND`, y si el fichero puede no existir, combinarla con `CREATE` (sin `CREATE`, da `NoSuchFileException`).

Operación	Código
Leer todas las líneas	`Files.readAllLines(ruta, charset)`
Leer en stream	`Files.lines(ruta)`
Leer todo como String	`Files.readString(ruta)`
Escribir líneas	`Files.write(ruta, lineas)`
Escribir un String	`Files.writeString(ruta, "texto\n")`
Añadir al final	`Files.write(ruta, lineas, StandardOpenOption.APPEND)`
Añadir creando si no existe	`Files.write(ruta, lineas, StandardOpenOption.CREATE, StandardOpenOption.APPEND)`

Cerrar un fichero

`Files` no tiene un método `close()`. Cerrar es una operación de los objetos que abres (`BufferedReader`, `BufferedWriter`, `InputStream`, `OutputStream`, canales, `Stream<String>`...), porque todos implementan la interfaz `AutoCloseable` / `Closeable`, que tiene el método `close()`.

Hay 3 formas de hacerlo:

Forma	Cómo	Comentario
No hace falta cerrar	`Files.readAllLines`, `Files.write`, `Files.writeString`, `Files.readString`...	Abren y cierran el fichero ellos solos
try-with-resources (recomendada)	`try (BufferedReader br = Files.newBufferedReader(ruta)) { ... }`	Se cierra automáticamente al salir del bloque, incluso si hay excepción
Manual	`br.close();`	Hay que ponerlo en un `finally`; es más propenso a olvidos

▼ Ejemplo con CSV

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.nio.file.StandardOpenOption;

public class EjemploCSV {

    public static void main(String[] args) {
        Path csv = Paths.get("clientes.csv");

        // LEER: el BufferedReader se cierra solo al salir del try
        try (BufferedReader br = Files.newBufferedReader(csv, Stand
ardCharsets.UTF_8)) {
            String linea;
            while ((linea = br.readLine()) != null) {
                String[] campos = linea.split(";");   // ; es el se
parador típico en CSV español
                System.out.println(campos[0] + " -> " + campos[1]);
```

```
            }
        } catch (IOException e) {
            System.out.println("Error leyendo el fichero: " + e.get
Message());
        }

        // AÑADIR una línea al final: el BufferedWriter también se
cierra solo
        try (BufferedWriter bw = Files.newBufferedWriter(csv, Stand
ardCharsets.UTF_8,
                StandardOpenOption.CREATE, StandardOpenOption.APPEN
D)) {
            bw.write("7;Ana;Madrid");
            bw.newLine();
        } catch (IOException e) {
            System.out.println("Error escribiendo en el fichero: "
+ e.getMessage());
        }
    }
}
```

▼

Tratamiento de ficheros de texto sobre “Beans” JAVA dao con clases de java.nio

▼ Reestructuración del proyecto

Partiendo del proyecto de la actividad 1.7 he sustituido las clases de `java.io` por las de `java.nio`, manteniendo las mismas funcionalidades. Las clases `GestorMapaMundi`, las interfaces DAO y los beans no han cambiado porque no usan `java.io`. Los cambios están en estas clases:

`GestorPropiedades`	`FileInputStream` pasa a `Files.newBufferedReader(Path, Charset)` con `try-with-resources`
`ContinenteDaoFichero`	Es la que más ha cambiado (ver tabla siguiente)
`PaisDaoFichero`	Lectura con `Files.readAllLines` y `Path`

Los cambios en `ContinenteDaoFichero` son:

`String FICHERO`	`Path FICHERO = Paths.get(...)` y nuevo atributo `Charset CHARSET`
`new BufferedReader(new FileReader(...))` y bucle `while`	`Files.readAllLines(FICHERO, CHARSET)` y bucle `for`

`(readLine)`	
`new BufferedWriter(new FileWriter(FICHERO, true))` para añadir	`Files.writeString(..., StandardOpenOption.CREATE, StandardOpenOption.APPEND)`
`new BufferedWriter(new FileWriter(FICHERO))` para reescribir	Se construye una `List<String>` y se escribe con `Files.write(FICHERO, lineas, CHARSET)`
`bw.close()` manual (y `BufferedReader` sin cerrar)	No hace falta: los métodos de `Files` abren y cierran el fichero

Al ejecutar por primera vez apareció un error de "Error al cargar el archivo de propiedades: Input length = 1". La causa fue que `application.properties` está guardado en ISO-8859-1 y yo lo leía como UTF-8, de modo que las propiedades no se cargaban y `Paths.get` recibía null.

▼ Análisis de código con IA

Le pasé el contenido de `ContinenteDaoFichero` a la IA con la pregunta: "¿Si hubiera que realizar 3 cambios sobre ese código, qué cambios realizarías?". La respuesta fue:

#	Cambio propuesto	Problema que resuelve
1	Sustituir los textos fijos (`"anyo"` , `"2026"` y los nombres de columna) por **constantes**	Estos literales se repiten en varios métodos; si hay que cambiarlos, hay que tocar muchos sitios
2	**Conservar el año original** de cada fila en lugar de escribir siempre `"2026"`	El año está fijo en el código. Con mi CSV no se nota porque todas las filas son 2026, pero si hubiera otros años se sobrescribirían al modificar o eliminar
3	**Mejorar el manejo de errores**: comprobar que cada línea tiene 3 campos antes de acceder a `trozos[1]` y `trozos[2]`	Una línea mal formada provoca `ArrayIndexOutOfBoundsException`

▼ Cambio de constantes

Elegí el primero porque es el más sencillo, no añade métodos nuevos y no cambia el comportamiento del programa. Los literales estaban repetidos en 6 sitios y ahora se definen una sola vez al principio de la clase.

La ventaja es que si cambia el año o el nombre de una columna ,se modifica en un único sitio en vez de seis. Después del cambio volví a ejecutar las pruebas y el resultado fue el mismo, así que la funcionalidad se mantiene.

```
public class ContinenteDaoFichero implements IContinenteDAO {

    private final Path FICHERO = Paths.get(GestorPropiedades.get("f
```

```
ichero.ruta.continentes"));
    private final String SEPARADOR = GestorPropiedades.get("ficher
o.separador.continentes");
    private final Charset CHARSET = StandardCharsets.UTF_8;

    private static final String PREFIJO_CABECERA = "anyo";
    private static final String ANYO_ACTUAL = "2026";
    private static final String COL_CODIGO = "codigo_continente";
    private static final String COL_NOMBRE = "nombre_continente";

    public Continente obtenerContinentePorCodigo(String codigo) {
    ...
```