Actividad 1.8: Tratamiento ficheros CVS – DAO properties – java.nio

A. Investigación sobre las clases del paquete java.nio.

Diferencias principales entre java.nio y java.io

1. **Orientación a Buffers vs. Streams: java.io trabaja orientado a flujos de datos (*Streams* de bytes o caracteres) de forma secuencial. java.nio trabaja orientado a Buffers y Channels, lo que permite mover datos en bloques e ir hacia adelante y hacia atrás en el buffer.**
2. **I/O Bloqueante vs. No Bloqueante: java.io es principalmente bloqueante (un hilo se detiene hasta que la operación de lectura/escritura termina). java.nio permite operaciones no bloqueantes (usando *Selectors* y *Channels*).**
3. **Manejo de rutas y archivos: java.io utiliza la clase java.io.File, mientras que java.nio.file introduce Path (interfaz) y Files (clase con operaciones utilitarias estáticas y atómicas para el sistema de archivos).**

Clases y métodos involucrados por operación (java.nio.file)

- **Crear un fichero/directorio:**
    - **Crear fichero: Files.createFile(Path path)**
    - **Crear directorio: Files.createDirectory(Path path) o Files.createDirectories(Path path) (crea carpetas intermedias si no existen).**
- **Borrar un fichero/directorio:**
    - **Files.delete(Path path) (lanza excepción si no existe) o Files.deleteIfExists(Path path).**
- **Renombrar un fichero/directorio:**
    - **En java.nio, el renombrado es una operación de movimiento en la misma ruta:**
    - **Files.move(Path origen, Path destino)**
- **Mover un fichero/directorio:**
    - **Files.move(Path origen, Path destino, CopyOption... options) (usando p. ej. StandardCopyOption.REPLACE_EXISTING).**

- **Abrir un fichero/directorio:**
    - **Fichero: Files.newInputStream(Path path), Files.newOutputStream(Path path), Files.newBufferedReader(Path path) o Files.newBufferedWriter(Path path).**
    - **Directorio: Files.newDirectoryStream(Path dir).**
- **Indicar un juego de caracteres (Charset):**
    - **Se pasa como parámetro utilizando StandardCharsets.UTF_8, StandardCharsets.ISO_8859_1, etc.**
    - **Ejemplo: Files.newBufferedReader(path, StandardCharsets.ISO_8859_1) o Files.readAllLines(path, StandardCharsets.UTF_8).**
- **Leer todas las líneas de un fichero (como lista de Strings):**
    - **Files.readAllLines(Path path) o Files.readAllLines(Path path, Charset cs).**
    - ***Alternativa eficiente para archivos grandes:* Files.lines(Path path) (devuelve un Stream<String>).**
- **Escribir una línea o líneas en un fichero:**
    - **Files.write(Path path, Iterable<? extends CharSequence> lines, OpenOption... options) o Files.writeString(Path path, CharSequence csq).**
- **Añadir información a un fichero ya existente:**
    - **Usando la opción de apertura de anexo: Files.write(path, lineas, StandardOpenOption.APPEND) o Files.writeString(path, texto, StandardOpenOption.APPEND).**
- **Cerrar fichero:**
    - **Los métodos de conveniencia de Files (readAllLines, write, writeString) abren y cierran los recursos automáticamente.**
    - **Cuando se utilizan BufferedReader, BufferedWriter, InputStream, OutputStream o DirectoryStream, implementan AutoCloseable, por lo que se cierran explícitamente con .close() o mediante una estructura try-with-resources:**