David Torres Aranha

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

Investigación sobre las clases del paquete `java.nio`

Tratamiento de Ficheros utilizando las clases del paquete `java.nio` (en vez de las del paquete `java.io`). ¿Cuáles son las principales diferencias con las clases del paquete `java.io` para el tratamiento de ficheros?

Principales diferencias con las clases del paquete `java.io`:

- **Enfoque**: `java.io` está orientado a flujos (streams) bloqueantes. `java.nio` (NIO.2) está orientado a búferes y canales, permitiendo operaciones asíncronas y no bloqueantes.
- **Gestión de rutas**: `java.io` agrupa ruta y operaciones en la clase File. `java.nio` separa estos conceptos usando la interfaz Path para la ruta y la clase utilitaria Files para las operaciones.
- **Manejo de errores**: `java.io` suele devolver booleanos (true/false) ante fallos (como no poder borrar un archivo). `java.nio` lanza excepciones detalladas (ej. NoSuchFileException), facilitando el control de errores.
- **Metadatos**: `java.nio` ofrece soporte nativo avanzado para leer y modificar atributos de archivos (permisos, propietarios) y para trabajar con enlaces simbólicos.

Clases y métodos involucrados (usando principalmente Path y Files)

- Crear un fichero/directorio:
  - Fichero: `Files.createFile(Path path)`
  - Directorio: `Files.createDirectory(Path path)` o `Files.createDirectories(Path path)`
- Borrar un fichero/directorio:
  - `Files.delete(Path path)` o `Files.deleteIfExists(Path path)`
- Renombrar un fichero/directorio:
  - Se utiliza el método mover hacia el mismo directorio con otro nombre: `Files.move(Path origen, Path destino, StandardCopyOption.REPLACE_EXISTING)`
- Mover un fichero/directorio:
  - `Files.move(Path origen, Path destino, StandardCopyOption.REPLACE_EXISTING)`
- Abrir un fichero/directorio:
  - `Files.newBufferedReader(Path path)` (para texto), `Files.newInputStream(Path path)` (para bytes) o `Files.newByteChannel(Path path)`.

- Indicar que un fichero utilice un juego de caracteres concreto:
    - Se pasa el parámetro `Charset` a los métodos de lectura/escritura usando `StandardCharsets.ISO_8859_1` o `StandardCharsets.UTF_8`. Ejemplo: `Files.readAllLines(path, StandardCharsets.ISO_8859_1)`.
- Leer todas las líneas de un fichero (como lista de Strings):
    - `Files.readAllLines(Path path)`
- Escribir una línea, o líneas en un fichero:
    - `Files.write(Path path, Iterable<? extends CharSequence> lines)` (para listas) o `Files.writeString(Path path, CharSequence csq)` (Java 11+ para textos simples).
- Añadir información a un fichero ya existente:
    - Se utiliza el método write añadiendo la opción de apertura: `Files.write(Path path, lines, StandardOpenOption.APPEND)`
- Cerrar fichero:
    - Si se usan métodos de utilidad como `Files.readAllLines()` o `Files.write()`, se cierran solos. Si se abren flujos explícitamente, se debe invocar `.close()` sobre el flujo (ej. `reader.close()`), preferiblemente utilizando un bloque `try-with-resources` para el cierre automático.