Actividad 1.8 (java.nio) VI - CRUD + DAO + Properties (MapaMundi)

¿En qué se diferencian java.nio y java.io?

- **Cómo leen los datos:** java.io lee o escribe la información byte a byte o letra a letra de forma secuencial y un poco más lenta. En cambio, java.nio utiliza bloques y canales, lo que le permite mover bloques enteros de datos de golpe y hacer que todo vaya mucho más rápido.

- **Trabajar sin esperas (No bloqueante):** Java.nio permite que el programa mande la orden y siga haciendo otras tareas mientras tanto

- **Herramientas mucho más modernas:** A partir de Java 7, java.nio.file implemento clases como Path y Files, que sustituyeron a java.io.File. Ahora es mas comodo gestionar rutas, carpetas, permisos y ficheros con un par de líneas.

Clases y métodos que vas a usar en tu día a día

- **Crear un fichero o una carpeta:**
    - **Qué usas:** Las clases Files, Path y Paths.
    - **Cómo se hace:** Para un archivo usas Files.createFile(...) y si necesitas una carpeta (o varias por dentro de golpe), tiras de Files.createDirectory(...) o Files.createDirectories(...).

- **Borrar un fichero o una carpeta:**
    - **Qué usas:** Files y Path.

    - **Cómo se hace:** Con Files.delete(...) borras (si no existe, salta error) o con Files.deleteIfExists(...) se asegura y solo borra si existe el archivo.

- **Renombrar un fichero o carpeta:**
    - **Qué usas:** Files, Path y StandardCopyOption.
    - **Cómo se hace:** Se usa el método Files.move(...) pero añadiendo la opción StandardCopyOption.REPLACE_EXISTING. Si le dices que se mueva dentro de la misma ruta cambiando el nombre, hace un "renombrar".
- **Mover un fichero o carpeta:**
    - **Qué usas:** Files, Path y StandardCopyOption.
    - **Cómo se hace:** Similar que el anterior con Files.move(...), pero esta vez mueve la carpeta de destino.
- **Abrir un fichero:**
    - **Qué usas:** Files, Path y FileChannel.
    - **Cómo se hace:** Para priorizar el rendimiento y el procesamiento avanzado, es posible emplear canales mediante FileChannel.open(...). Por otro lado, si se prefiere mantener un enfoque tradicional adaptado a las nuevas interfaces, se puede recurrir a un lector optimizado utilizando Files.newBufferedReader(...).
- **Elegir el idioma o tipo de letras (Codificación):**
    - **Qué usas:** Charset y StandardCharsets.
    - **Cómo se hace:** Este permite la implementación de símbolos como pueden ser las ñ o tildes. Lo mas comun es usar StandardCharsets.UTF_8 o pedir una concreta como Charset.forName("ISO-8859-1").
- **Leer todo un fichero de golpe:**
    - **Qué usas:** Files, Path y StandardCharsets.
    - **Cómo se hace:** Con llamar a Files.readAllLines(path) te devuelve directamente una lista con todas las líneas del archivo en formato texto (por defecto usa UTF-8).

- **Escribir líneas en un fichero:**
    - **Qué usas:** Files, Path y StandardCharsets.
    - **Cómo se hace:** Utilizas Files.write(...) pasándole la ruta y la lista de textos que quieres guardar.
- **Añadir contenido al final de un fichero que ya existe:**
    - **Qué usas:** Files, Path y StandardOpenOption.
    - **Cómo se hace:** Usas Files.write(...) o un escritor por bloques añadiendo la opción StandardOpenOption.APPEND para que no se borre lo que ya había escrito antes.
- **Cerrar el fichero correctamente:**
    - **Qué usas:** La estructura try-with-resources.
    - **Cómo se hace:** Te ahorra colocar .close() munualmente al final. Al poner un bloque try (...) Java se encarga de cerrar el archivo automáticamente en cuanto termina de usarlo para no dejar fugas de memoria.