Métodos de Path Interface, Clase File, y Métodos Stream con NIO

1. Interfaz `Path`

La interfaz `Path` en `java.nio.file` representa una ruta de archivo o directorio en el sistema de archivos. Ofrece métodos para manipular y analizar rutas.

Método	Descripción	Ejemplo
Path getFileName()	Obtiene el nombre del archivo o directorio final en la ruta.	Path fileName = path.getFileName();
Path getParent()	Obtiene la ruta al directorio padre.	Path parent = path.getParent();
Path getRoot()	Devuelve la ruta raíz (ej., "C:\" en Windows).	Path root = path.getRoot();
Path resolve(Path p)	Combina la ruta actual con la ruta dada, creando una nueva ruta.	Path newPath = path.resolve("subdir/file.txt");
Path relativize(Path p)	Crea una ruta relativa desde la ruta actual a la ruta dada.	Path relative = path.relativize(path2);
boolean startsWith(Path p)	Verifica si la ruta empieza con la ruta dada.	boolean starts = path.startsWith("/home/user");
boolean endsWith(Path p)	Verifica si la ruta termina con la ruta dada.	boolean ends = path.endsWith("file.txt");
Path normalize()	Elimina elementos redundantes de la ruta, como "." o "..".	Path normalizedPath = path.normalize();
Path toAbsolutePath()	Convierte la ruta en una ruta absoluta.	Path absPath = path.toAbsolutePath();
boolean isAbsolute()	Verifica si la ruta es absoluta.	boolean absolute = path.isAbsolute();
Path subpath(int, int)	Obtiene un sub camino de la ruta especificada entre índices.	Path subPath = path.subpath(0, 2);

2. Clase `File`

La clase `File` representa un archivo o directorio y permite operaciones en el sistema de archivos.

Método	Descripción	Ejemplo
boolean exists()	Verifica si el archivo o directorio existe.	boolean exists = file.exists();
boolean isDirectory()	Verifica si es un directorio.	boolean isDir = file.isDirectory();
boolean isFile()	Verifica si es un archivo.	boolean isFile = file.isFile();
String getName()	Obtiene el nombre del archivo o directorio.	String name = file.getName();
String getPath()	Devuelve la ruta completa como `String`.	String path = file.getPath();
boolean mkdir()	Crea un directorio.	boolean dirCreated = file.mkdir();
boolean delete()	Elimina el archivo o directorio.	boolean deleted = file.delete();
String[] list()	Lista el contenido de un directorio como un arreglo de nombres.	String[] files = dir.list();
File[] listFiles()	Lista el contenido de un directorio como un arreglo de objetos `File`.	File[] files = dir.listFiles();
long length()	Obtiene el tamaño del archivo en bytes.	long size = file.length();

boolean renameTo(File dest)	Renombra el archivo o directorio al destino especificado.	boolean renamed = file.renameTo(new File("newname.txt"));
long lastModified()	Devuelve el tiempo de la última modificación en milisegundos desde el 1 de enero de 1970.	long lastModified = file.lastModified();

3. Métodos de Stream API con NIO

A través de `java.nio.file.Files`, la API de Streams facilita la lectura y manipulación de archivos mediante Streams.

Método	Descripción	Ejemplo
Stream<Path> list(Path dir)	Lista el contenido de un directorio como `Stream<Path>`.	try (Stream<Path> stream = Files.list(Paths.get("/dir"))){ ... }
Stream<String> lines(Path path)	Lee todas las líneas de un archivo como un `Stream<String>`.	try (Stream<String> lines = Files.lines(Paths.get("file.txt"))){ ... }
Stream<Path> walk(Path start, int depth)	Camina a través de un directorio, opcionalmente especificando la profundidad.	try (Stream<Path> stream = Files.walk(Paths.get("/dir"), 2)) { ... }
Stream<Path> find(Path start, int maxDepth, BiPredicate<Path, BasicFileAttributes>)	Busca archivos en un directorio según el predicado dado.	Stream<Path> found = Files.find(start, 2, (p, a) -> a.isRegularFile());
Stream<Path> lines(Path path, Charset cs)	Lee las líneas de un archivo con el conjunto de caracteres especificado.	try (Stream<String> lines = Files.lines(path, StandardCharsets.UTF_8)) { ... }