Functionality in Common

This slide shows you quite a bit of the functionality you'd expect from a File System, and the methods you'd use for each of these classes.

Functionality	**File instance methods**	**File static methods, with Path argument**
create file	`createNewFile()`	`createFile(Path p)`

delete directory of file	`delete()`	`delete(Path p)`
		`deleteIfExists(Path p)`

check path type	`isDirectory()`
	`isFile()`
		`isDirectory(Path p)`
		`isRegularFile(Path p)`

get byte size of file	`length()`	`size(Path p)`
List directory contents	`listFiles`	`list(Path p)`

create directory or directories	`mkdir()`
	`mkdirs()`
		`createDirectory(Path p)`
		`createDirectories(Path p)`

Rename	`renameTo(File dest)`	`move(Path src, Path dest)`

NIO2 file operations have been improved

The NIO2 types include support for:

- Asynchronous file I/O operations.
- File locking, including more granular locking. This means, instead of locking the entire file, a region of it can be locked.
- File metadata retrieval.
- Symbolic link manipulation.
- File system notifications. This means changes occurring on a path, can be made watchable to registered services.

NIO2 file operations are better performant

NIO2 types are non-blocking, meaning asynchronous access to resources, by multiple threads, is supported.

They manage memory more efficiently, reading and writing files directly to and from memory into buffers, through something called a FileChannel.

You can also read from or write to multiple buffers in a single operation.