Trail: Essential Classes
Lesson: Basic I/O
Section: File I/O (Featuring NIO.2)

The Java Tutorials have been written for JDK 8. Examples and practices described in this page don't take advantage of improvements introduced in later releases.

Checking a File or Directory

You have a `Path` instance representing a file or directory, but does that file exist on the file system? Is it readable? Writable? Executable?

Verifying the Existence of a File or Directory

The methods in the `Path` class are syntactic, meaning that they operate on the `Path` instance. But eventually you must access the file system to verify that a particular `Path` exists, or does not exist. You can do so with the `exists(Path, LinkOption...)` and the `notExists(Path, LinkOption...)` methods. Note that `!Files.exists(path)` is not equivalent to `Files.notExists(path)`. When you are testing a file's existence, three results are possible:

- The file is verified to exist.
- The file is verified to not exist.
- The file's status is unknown. This result can occur when the program does not have access to the file.

If both `exists` and `notExists` return `false`, the existence of the file cannot be verified.

Checking File Accessibility

To verify that the program can access a file as needed, you can use the `isReadable(Path)`, `isWritable(Path)`, and `isExecutable(Path)` methods.

The following code snippet verifies that a particular file exists and that the program has the ability to execute the file.

```
Path file = ...;
boolean isRegularExecutableFile = Files.isRegularFile(file) &
         Files.isReadable(file) & Files.isExecutable(file);
```

Note: Once any of these methods completes, there is no guarantee that the file can be accessed. A common security flaw in many applications is to perform a check and then access the file. For more information, use your favorite search engine to look up `TOCTTOU` (pronounced *TOCK-too*).

Checking Whether Two Paths Locate the Same File

When you have a file system that uses symbolic links, it is possible to have two different paths that locate the same file. The `isSameFile(Path, Path)` method compares two paths to determine if they locate the same file on the file system. For example:

```
Path p1 = ...;
Path p2 = ...;

if (Files.isSameFile(p1, p2)) {
    // Logic when the paths locate the same file
}
```


Previous page: File Operations
Next page: Deleting a File or Directory