File System In Java

Main Sources:

<https://docs.oracle.com/javase/8/docs/api/java/io/File.html>

<https://docs.oracle.com/javase/8/docs/api/java/io/IOException.html>

* Creating A Folder In Java:

The **mkdir()** method is a part of File class. The mkdir() function is used to create a new directory denoted by the abstract pathname. The function returns true if directory is created else returns false.

Example Code:

File directory = new File(“Directory");

if (directory.mkdir()) {

System.out.println("Directory created");

} else {

System.out.println("Failed to create directory");

}

Source: [geeksforgeeks](https://www.geeksforgeeks.org/file-mkdir-method-in-java-with-examples/)

* Creating A File In Java:

The **createNewFile()** function is a part of File class in Java . This function creates new empty file. The function returns true if the abstract file path does not exist and a new file is created. It returns false if the filename already exists.

Example Code:

File file = new File("myFile.txt");

try {

if (file.createNewFile()) {

System.out.println("File created");

} else {

System.out.println("File already exists");

}

} catch (IOException e) {

e.printStackTrace();

}

Note: try-catch goes into more detail about how if the file cannot be created java uses the exception (IOException) and without the try-catch from what i seem to understand, the program would simply crash, it has been used in my code but i still don’t completely understand the concept, but overall from what I understand so far it’s a failsafe method for issues during file creation (no permission, invalid path, memory is full) and as long as there are no issues with the try part of the code catch will never be executed

Source:

[Geeksforgeeks](https://www.geeksforgeeks.org/file-createnewfile-method-in-java-with-examples/) (where try-catch was seen getting used multiple times)

[Exception Handling Video Tutorial](https://www.youtube.com/watch?v=1XAfapkBQjk) 0:00 - 6:00

ClaudeAI

* Checking If File/Folder exists:

You can use **File.exists()** to test existence of either a file or directory.

You can also use **java.nio.file.Files.exists()** but it seems to use a completely different library and also on [stackoverflow forums](https://stackoverflow.com/questions) there have been mixed opinions with some saying its a lot faster that java.io.File.exists(), others saying it’s 5 times slower

Example Code:

public void checkPath(String name) {

File target = new File(currentDirectory, name);

if (target.exists()) {

if (target.isDirectory()) {

System.out.println(name + " is a directory");

} else if (target.isFile()) {

System.out.println(name + " is a file");

}

} else {

System.out.println(name + " does not exist");

}

}

Source:

[Oracle Docs](https://docs.oracle.com/javase/tutorial/essential/io/check.html)

[Stackoverflow](https://stackoverflow.com/questions/1816673/how-do-i-check-if-a-file-exists-in-java)

* View Contents Of A Folder:

The **listFiles()** method asks Java to fetch every item inside the current folder and then prints each item’s name one by one.

Example Code:

public void listDirectory() {

File[] items = currentDirectory.listFiles();

if (items != null) {

for (File f : items) {

System.out.println(f.getName());

}

}

}

Sources:

[Geeksforgeeks](https://www.geeksforgeeks.org/file-listfiles-method-in-java-with-examples/)

[Oracle Docs](https://docs.oracle.com/javase/8/docs/api/java/io/File.html)

ClaudeAI

* The Difference Between File And Path:

As confusing as this one gets there i was unable to find a direct explanation that on the internet, from what it seems

**File:** is the legacy API **(java.io.File)** used for handling file and directory names with built in I/O methods [mkdir(), listFiles(), createNewFile()]

It also usually returns a boolean, true or false according to the success or failure of the operation

It cannot handle symbolic links

**Path:** is newer and was added with the introduction of Java 7 and it was introduced with a newer API **(java.nio.file.Path)** which is everything java.io.File is and more and it more flexible and it can do more operations with more efficiency

On failure it provides more detailed error information

It can handle symbolic links

But both of them could be converted to the other using **file.toPath()** for files and **path.toFile()** for paths

Sources

[stackoverflow](https://stackoverflow.com/questions/32143633/java-io-file-vs-java-nio-files-which-is-the-preferred-in-new-code)

[Baeldung](https://www.baeldung.com/java-path-vs-file)

[Oracle Docs / Path](https://docs.oracle.com/javase/8/docs/api/java/nio/file/Path.html)

[Oracle Docs / File](https://docs.oracle.com/javase/8/docs/api/java/io/File.html)

ChatGPT - Organising the differences