Java → File processing → File hierarchies

Theory: File hierarchies

🕓 20 minutes 0 / 5 problems solved Skip this topic Start practicing

1109 users solved this topic. Latest completion was about 10 hours ago.

As you know, an application can store data in files on a disk. To structure the data, files can be organized into directories. One directory, ***a parent***, can include other directories, ***subdirectories***. This is how a hierarchy of files is created. For example, consider the filesystem hierarchy in Linux: it has the root directory `/` that includes all other files and directories, even if they are stored on different physical devices.

§1. Methods for Iterating through file hierarchies

It is possible to iterate through a file hierarchy in Java programs by using the `java.io.File` class. Let's consider four methods for doing it:

- `File getParentFile()` returns an instance of `java.io.File` representing the parent directory of this file, or `null` if this file does not have the parent (it means it is the root);
- `String getParent()` returns a string representation of the parent directory of this file, or `null` if this file does not have a parent;
- `File[] listFiles()` returns an array of files that are located in this directory, or `null` if this instance is not a directory;
- `String[] list()` returns an array of strings naming the files and directories in the directory denoted by this abstract pathname.

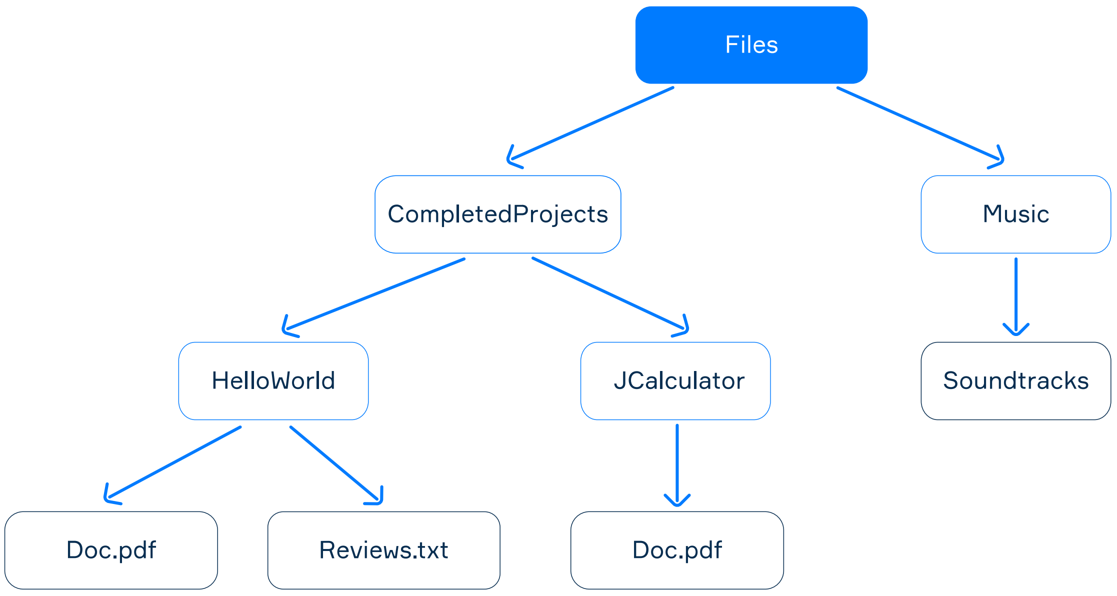
Any instance of `java.io.File` has all these methods.

In the next section, we'll find out how to apply these methods while dealing with a simple hierarchy.

§2. A hierarchy example

Let's consider a file hierarchy with a root directory named `Files`. It contains two subdirectories: `CompletedProjects` and `Music`. They also have subdirectories. The `HelloWorld` directory contains two files related to the project, the `JCalculator` directory contains only one file and the `Soundtracks` directory is empty.

Take a look at the illustration of this hierarchy below.



An example of a file hierarchy

You can reproduce this hierarchy in your file system to try and execute the following examples.

Let's suppose we have an instance of `java.io.File` named `completedProjsDirectory`. It corresponds to the `CompletedProjects` directory. Let's now get both of its subdirectories, containing data on projects.

Current topic:

File hierarchies ···

Topic depends on:

✓ NPE ···

✗ Files ···

Table of contents:

```
File[] projects = completedProjsDirectory.listFiles(); // HelloWorld and JCalculator
```

The particular order of files in the array is not guaranteed. To find the `HelloWorld` project we will use a special custom method called `findFileByName` that returns either a found file or `null`.

```
File helloWorldProject = findFileByName(projects, "HelloWorld"); // just find a file
```

We assume that it is not `null` just to simplify the code for education purposes. Still, it is better to check because the actual hierarchy may be changed.
Let's now get the `Reviews.txt` file by using the same method.

```
File reviews = findFileByName(helloWorldProject.listFiles(), "Reviews.txt");
```

We assume here that it is found (but keep NPE in mind in your future work). Now let's try to get a list of files from our current File instance.

```
File[] files = reviews.listFiles(); // null
```

The array list is `null`, because `reviews` is not a directory at all and cannot include other files or subdirectories.

Now let's go back to the `CompletedProjects` directory to try and get its parent named `Files`.

```
File filesDirectory = completedProjsDirectory.getParentFile();
```

In the following code, we get the directory `Soundtracks` (skipping again all null checking).

```
File music = findFileByName(filesDirectory.listFiles(), "Music");
File soundtracks = findFileByName(music.listFiles(), "Soundtracks");
```

The directory does not include other files, so the method `listFiles()` returns the empty array.

```
int length = soundtracks.listFiles().length; // 0, but not null
```

As you can see, it returns 0 but not `null` as in the case when the instance is not a directory.

§3. Conclusion

We've learned how directories can be included in each other, forming a hierarchy of files. We've considered several methods for file hierarchies traversal working on an example hierarchy. However, these methods allow developing general algorithms to traverse any file hierarchy. We skipped the `null` checking to simplify the example, but do not forget about it to avoid NPE in your programs.

Report a typo

97 users liked this theory. 5 didn't like it. **What about you?**

😍 🙂 😐 🙁 😡

Start practicing

Comments (7) Hints (0) Useful links (0) Show discussion