Quiz yourself: Use the Files class to copy or move files and directories

JAVA SE

Quiz yourself: Use the Files class to copy or move files and directories

Test your knowledge of the java.nio.file.Files class.

by Simon Roberts and Mikalai Zaikin

October 19, 2020

If you have worked on our quiz questions in the past, you know none of them is easy. They model the difficult questions from certification examinations. We write questions for the certification exams, and we intend that the same rules apply: Take words at their face value and trust that the questions are not intended to deceive you but to straightforwardly test your knowledge of the ins and outs of the language.

The objective of this quiz is to determine what you know about using the `java.nio.file.Files` class and how it works with files and directories.

Which of the following statements is true? Choose one.

A. It cannot move a nonempty directory. The answer is A.

B. It cannot copy a file if a target file with the same name already exists. The answer is B.

C. It can copy a file to a nonexistent folder and the required target folder tree will be created automatically. The answer is C.

D. It can move a file atomically. The answer is D.

Answer. The `java.nio.file.Files` class in Java 11 provides a wide variety of `static` utility methods for working with files and directories. Two particular methods from this group are discussed in this question. The following methods copy and move a file in the file system:

```
Files.copy (Path source, Path target, CopyOpt
Files.move (Path source, Path target, CopyOpt
```

By the way, the `Files` class contains *three* overloaded methods called `copy`, but the other two work with input/output stream data.

The `Files.move()` method has some restrictions related to directories, which are described in its documentation as follows: “This method may be invoked to move an empty directory.”

The documentation then continues to constraints on moving nonempty directories, which can *sometimes* be moved. The ability to move a nonempty directory is based on whether the source and target are in the same `java.nio.file.FileStore`, which might correspond, for example, to being on the same disk. Given these observations, option A is incorrect.

The documentation for the copy method notes that, by default, the `Files.copy()` method throws a `java.nio.file.FileAlreadyExistsException` exception if a target file already exists. However, the developer has the option to alter this behavior by adding `StandardCopyOption.REPLACE_EXISTING` as one of the varargs parameters to the method invocation. In this case, if the target file exists, it will be overwritten. From this, you can see that option B is incorrect.

Option C suggests that missing directories leading to the target of a copy operation will be created automatically. This is not obvious from the documentation, but if you try it, you’ll find that it will fail. In this situation, the method will throw a `java.nio.file.NoSuchFileException` exception. In view of this, you can see that option C is incorrect.

Note: The documentation for the `Files` utility class describes the `createDirectories` method by noting that it “Creates a directory by creating all nonexistent parent directories first.” Clearly this would be a helpful method for addressing the goal implicit in option C, that is, copying a file to a nonexistent folder and creating the needed directory tree automatically.

What about option D? The documentation for the `move` method describes the `StandardCopyOption.ATOMIC_MOVE` option, which can be used to move a file atomically. However, as with option A, this will work only within the same `FileStore`. If you try to move atomically from one `FileStore` to another, the method will throw an exception: `java.nio.file.AtomicMoveNotSupportedException`. In that case, the move operation can succeed if you don’t require the atomic behavior. Of course, such a move will be slightly less reliable since it might copy some of the files and then fail, resulting in an unpredictable final state on the file system. So, don't do that.

Conclusion: The correct answer is option D.

Simon Roberts

Simon Roberts joined Sun Microsystems in time to teach Sun's first Java classes in the UK. He created the Sun Certified Java Programmer and Sun Certified Java Developer exams. He wrote several Java certification guides and is currently a freelance educator who publishes recorded and live video training through Pearson InformIT (available direct and through the O'Reilly Safari Books Online service). He remains involved with Oracle's Java certification projects.

Mikalai Zaikin

Mikalai Zaikin is a lead Java developer at IBA IT Park in Minsk, Belarus. During his career, he has helped Oracle with development of Java certification exams, and he has been a technical reviewer of several Java certification books, including three editions of the famous *Sun Certified Programmer for Java* study guides by Kathy Sierra and Bert Bates.

Share this Page