NOTE: SCENARIO IS WHAT THE PUZZLE TAKER SEES.

SCENARIO:

You are developing an application for managing files and directories. The application requires data on the average size of the files in each directory (excluding its subdirectories). The following code implements the `averageFileSize` method for this computation. The `averageFileSize` method takes one argument `path` (a `String` representing the path of a directory or file) and calculates and returns the average size of files in the given path if the given path is a directory, or returns the size of the file if the given path is a file. Consider the snippet of code below and answer the following questions, assuming that the code has all required permissions to execute.

Code:

```
01  // OMITTED: Import whatever is needed.
02  public final class DirectoryUtils {
03    public static long averageFileSize (String path) {
04      // OMITTED: Acquire a FileLock object to prevent other
05      // processes from changing the path content while calculating
06      // the average file size. Assume nothing goes wrong here.
07
08      int count = 0;
09      long total = 0;
10      File file = new File(path);
11      if (file.isDirectory()) {
12        File[] files = file.listFiles();
13        if (files != null) {
14          for (File child : files)
15            if (child.isFile()) {
16              count++;
17              total += child.length();
18            }
19        }
20      }
21      else if (file.isFile()) {
22        total = file.length();
23      }
24
25      // OMITTED: Release properly the FileLock object
26      // acquired at the beginning of the method.
27
28      if (count != 0)
29        return (total / count);
30      else
31        return total;
32    }
33  }
```

Questions:
1. What will the `averageFileSize` method do when executed?

2. Which one statement is correct if the `averageFileSize` method gets called with a non-null, non-empty path as its argument?

a. The method always returns an integer value (of type `long`).
b. The method throws an exception on line 12, in case of any failure while getting the list of the directory content.
c. The method throws an exception on lines 16 or 20, in case of any failure while reading the size of the file.
d. The method throws an exception on a line other than 12, 16, or 20, in case of any failure while getting the list of the directory content.
e. None of the above.

[Other statistical questions will be imported here while creating the survey.]

NOTE: ANSWER IS TO BE SHOWN TO THE PUZZLE TAKER AT THE END OF THE SESSION.

ANSWER:
a
According to the API documentation, none of the methods used in the code throw an exception.

NOTE: THE REST OF THE DOCUMENT CONTAINS EXTRA INFORMATION FOR THE PROJECT RESEARCHERS. IT IS NOT TO BE SHOWN TO THE PUZZLE TAKERS.

TAGS:

java, file-class, io-operation, file-operation, null-pointer-exception, deprecated-api

CATEGORIES:
Blindspot - NO
Type - File
Number of distinct functions - 6
Number of total functions - 8
Blindspot function - `NA`
Function call omitted - NA
Blindspot type - NA
Number of Parameters in the blindspot function - NA
Cyclomatic complexity - 6

NAME:
File class, list, listFiles method - An abstract representation of file and directory pathnames.

MORE INFORMATION:
To see an incorrect way to use the API, look at the J15-File.listFiles puzzle.

REFERENCES:
1. File