NOTE: SCENARIO IS WHAT THE PUZZLE TAKER SEES.

SCENARIO:

Consider an application that writes some information of its users to a file when users log in to the system. The application has one (and only one) "user information manager" (an object of type `UserInfoFileManager`) that handles creation and deletion of the file containing user information. As the file contains sensitive information, it must be deleted when the application terminates. In the following implementation of the class, the file is created when an instance of the class is created (in the constructor) and is deleted in the `finalize` method (which is to be called by the garbage collector on the "user information manager" when garbage collection determines that there are no more references to it.), 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 UserInfoFileManager {
03    private final File file;
04
05    /**
06     * Default constructor, initialize the "file" member
07     */
08    public UserInfoFileManager () {
09      file = new File("users.info");
10    }
11
12    /**
13     * This method writes the given information into the file.
14     * @param information the information to be written
15     * @throws IOException IOException thrown during writing
16     */
17    public final void write (String information)
18        throws IOException {
19      try (FileOutputStream output = new FileOutputStream(file)) {
20        output.write(information.getBytes());
21      }
22    }
23
24    // Delete the file during the object finalization
25    public void finalize () throws Throwable {
26      file.delete();
27      super.finalize();
28    }
29  }
```

Questions:
1. What will happen when an instance of the class is created and then destroyed?

2. Considering the file pointed by the `file` member, and assuming that the application is the only process having access to the file, which one statement is correct?
a. The file might be deleted (via the `finalize` method call) while the application is writing to it (via the `write` method call).
b. The file is always deleted when the application terminates.
c. The file is never deleted.
d. It is not possible to do I/O operations in the `finalize` method.
e. None of the above

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

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

ANSWER:
e
The file may or may not be deleted. The `finalize` method is called when an object is about to get garbage collected. That can be at any time after the object ("user information manager" in this case) has become eligible for garbage collection. Note that it is entirely possible that an object never gets garbage collected. This can happen when the object never becomes eligible for garbage collection or when no garbage collection actually runs between the time the object become eligible and the time the JVM stops running.

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

TAGS:
java, object-finalization, garbage-collector

CATEGORIES:
Blindspot - YES
Type - GC (Garbage Collection)
Number of distinct functions - 5
Number of total functions - 5
Blindspot function - `Object.finalize();`
Function call omitted - NO
Blindspot type - NA
Number of Parameters in the blindspot function - 0 parameters
Cyclomatic complexity - 1

NAME:
Object class, finalize method - Called by the garbage collector on an object when the object is eligible for garbage collection.

DESCRIPTION:
It is common in applications to store sensitive information in a temporary storage for further processing. In such situations the information must be cleaned up as soon as they are not needed anymore.

BLINDSPOT:
The general contract of the `finalize` method is that it is invoked if and when JVM has determined that there is no longer any means by which this object can be accessed by any thread that has not yet died. However, that it is entirely possible that an object never gets garbage collected, thus the `finalize` method never executes.

CORRECT USE EXAMPLE:
#N/A

MORE INFORMATION:
#N/A

REFERENCES:
1. Object#finalize