NOTE: SCENARIO IS WHAT THE PUZZLE TAKER SEES.

SCENARIO:

You are developing a secure texting system. Text messages must be encrypted before being transferred over the network. The `encrypt` method takes three arguments: `alg`, which is a `String` value indicating the encryption algorithm, `key`, which is a `Key` object to be used as the encryption key, and `text`, a `String` value to be encrypted. The method returns a `byte` array representing the ciphertext of the original (plain) text. Consider the snippet of code below and answer the following questions, assuming that the code has all required permissions to execute.

```
01   // OMITTED: Import whatever is needed.
02   public final class CryptoUtils {
03     public static byte[] encrypt (String alg, Key key, String text)
04         throws GeneralSecurityException {
05       if (text == null)
06         return null;
07
08       if (text.isEmpty())
09         return new byte[0];
10
11       // Create a cipher
12       Cipher cipher = Cipher.getInstance(alg);
13       cipher.init(Cipher.ENCRYPT_MODE, key);
14
15       // Encrypt the data
16       byte[] encrypted = cipher.update(text.getBytes());
17       return encrypted;
18     }
19   }
```

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

2. Which one statement is correct if the `encrypt` method gets called to encrypt an arbitrary (non-null, non-empty) text with arbitrary (yet valid) algorithm and `key` object?

a. The `encrypt` method always encrypts the text correctly.
b. The `encrypt` method never encrypts the text correctly, and it throws an exception.
c. The `encrypt` method never encrypts the text correctly, but it does not throw an exception.
d. Depending on the encryption algorithm, the `encrypt` method may encrypt the text correctly.
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:
d
According to the API documentation it's better to call the `doFinal` method at the end of processing of data. For some algorithms it is a must. The `doFinal` method is very similar to the `update` method in that it may also put out 0 or more bytes, depending on the encryption algorithm. Otherwise the encrypted data is not valid and may not be decrypted properly. Thus, even though the `encrypt` method does not throw any exception, the produced data (in this case ciphertext) is invalid and cannot be decrypted to the original text. Depending on the encryption algorithm, this code might work in some cases, though it is not generally the correct way of using cipher objects.

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, cryptography, cipher, finalization, api-protocol-usage

CATEGORIES:
Blindspot - YES
Type - Crypto
Number of distinct functions - 5
Number of total functions - 5
Blindspot function - `Cipher.doFinal()`
Function call omitted - YES
Blindspot type - Omitted function call
Number of Parameters in the blindspot function - 2 parameters
Cyclomatic complexity - 3

NAME:
Cipher class, doFinal method - Provides the functionality of a cryptographic cipher for encryption and decryption. It forms the core of the Java Cryptographic Extension (JCE) framework.

DESCRIPTION:
The creation and use of a `Cipher` object follows a simple pattern: You create one using the `Cipher.getInstance` method, initialize it with the mode you want using the `init` method, feed the input data in while collecting output at the same time using the `update` method, and then finish off the process with the `doFinal` method.

BLINDSPOT:
The API usage protocol specified by the Java standard library (i.e. how to use cipher objects) may cause a blindspot, especially because, in some cases, violating the protocol will have no implication. In addition, codes violating the protocol do not cause any error or exception. Instead, they silently produce invalid data. For example, if one tries to encrypt a plaintext using the given code in the scenario, the code generates an invalid ciphertext, meaning the generated ciphertext cannot be decrypted to the original plaintext.

CORRECT USE EXAMPLE:

Code:
```
import javax.crypto.*;
public class Application {
    public static void main ( String... args ) throws Exception {
        byte[] data = …; // Get some data to encrypt

        SecretKey key = …; // Create a key!
        Cipher c = Cipher.getInstance(…); // Create a cipher
        c.init(Cipher.ENCRYPT_MODE, key);

        System.out.println("Start encryption...!");

        byte[] encrypted = new byte[c.getOutputSize(data.length)];
        int length = c.update(date, 0, data.length, encrypted, 0);
        length += c.doFinal(encrypted, length);

        System.out.println("Done!");
    }
}
```

MORE INFORMATION:
#N/A

REFERENCES:
1. Cipher