NOTE: SCENARIO IS WHAT PUZZLE TAKER SEES.

SCENARIO:

You are developing a secure texting system. The software requirement for this system is to encrypt text messages before transferring them over network. The `encrypt` method takes three arguments: `String alg` (an encryption algorithm), `Key key` (a cryptographic key used for encryption), `String text` (a string value to be encrypted), and 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
06       // Create a cipher
07       Cipher cipher = Cipher.getInstance(alg);
08       cipher.init(Cipher.ENCRYPT_MODE, key);
09
10       // Encrypt the data
11       byte[] bytes = text.getBytes();
12       byte[] output = new byte[cipher.getOutputSize(bytes.length)];
13       cipher.update(bytes, 0, bytes.length, output, 0);
14       cipher.doFinal(output);
15
16       return output;
17     }
18   }
```

Questions:

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

2. If the `encrypt` method gets called with a valid algorithm and valid key and a non empty text string, which one statement is correct?

a. Depending on the key, the `encrypt` method may encrypt the text correctly.
b. Depending on the algorithm, the `encrypt` method may encrypt the text correctly.
c. The `encrypt` method never encrypts the text correctly.
d. The `encrypt` method always encrypts the text correctly.
e.The encrypt method will lead to a crash in the program where it is invoked.

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

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

ANSWER:
b
The code snippet, as is, does not encrypt the data properly. According to the API documentation the `doFinal(byte[] input)` method (called in the above code line #14) takes the argument as input data, not output. Thus the method `doFinal` does not do the finalization on the output, but it treats the argument as input data and encrypts it and returns the encrypted version of the argument. The correct usage of the method is to pass both the array of encrypted data and the offset returned by the previous call of the `update` method.

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

TAGS:
java, cryptography, cipher, ignoring-return-value, api-protocol-usage

CATEGORIES:
Blindspot - YES
Type - Crypto
Number of distinct functions - 7
Number of total functions - 7
Blindspot function - `Cipher.update()`
Function call omitted - NO
Blindspot type - Return type ignored
Number of Parameters in the blindspot function - 4 parameters
Cyclomatic complexity - 2

NAME:
Cipher class, update 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. If the data is going to be processed in more than one step, i.e. calling the `update` method more than once (in this context the `doFinal` method is considered as the `update` method), the return value of the `update` method should be saved and passed as `offset` to the next call.

BLINDSPOT:

According to the API documentation, ignoring the value returned by the `update` method and causes invalid data processing.

CORRECT USE EXAMPLE:

```
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