CSC 110: Fundamentals of Programming I
Assignment #3: Code testing, parameters, return statements

Due date

Sunday, October 4th, 2015 at 11:55 pm via submission to connex.

How to hand in your work

Submit the requested file (`Encode.java`) through the Assignment #3 link on the CSC 110 connex site.

Learning outcomes

When you have completed this assignment, you should understand:

- How to write *tests* to ensure that code runs as intended.
- How to write and call a *static method* that receives *parameters*.
- How to *return* a value from a *static method*.
- How to use *String methods*.
- How to *indent and document* a Java program.

Download and update the Java program named *Encode.java*. The program includes a method named *encrypt* that accepts a `String` and an `int` as parameters. The method will return a new encoded `String` by adding the integer value to each character's ASCII value, as shown in the examples and following algorithm below.

Examples:

1) String: HELLO key: 2

```
H + 2 = J
E + 2 = G
L + 2 = N
L + 2 = N
O + 2 = Q
```

Resulting String: JGNNQ

2) **String: TEST key: 11**

```
T + 11 = E
E + 11 = P
S + 11 = D
T + 11 = E
```

Resulting String: EPDE

Note: T + 6 would result in Z. Adding 11 to T went past the value of Z, with 5 remaining. In this case the remaining 5 are counted beginning at the start of the alphabet, resulting in E.

Algorithm:

String **message**, int **key** are passed in as parameters.

1. Create a new empty String **encodedMessage**
2. Convert the String to upper-case.
3. For each character in **message**
    a. Convert each character to its integer value
    b. Assign this value to an int **characterValue**
    c. Add **key** to **characterValue**. Make sure the new value is within the acceptable range**
    d. Convert **characterValue** back to a character and add it to the end of **encodedMessage**
4. Return **encodedMessage**

** Work on getting the encrypt method working for simple examples first (like example 1 above) before working on more difficult examples (like example 2, where the letters wraps around from Z back to the start of the alphabet)

In your program, you will fill in the code for 3 methods

1. A static method called *testing()*, which does the following:
    a. Creates a `String` to be used as the message to be encoded.
    b. Creates an `int` to be used as the encryption key for the encoding.
    c. Creates another `String` to hold the result of the encoding.
    d. Tests the *encode* method with multiple messages and encryption keys

Note: The example on the previous page shows that the message: `HELLO` with key: `2` results in the encoded message: `JGNNQ`. You must test that this is true in your solution. Another recommended follow-up test might be to make sure that the message: `JGNNQ` with key: `-2` results in the encoded message: `HELLO`. The encrypt method must be completed before these tests will work.

2. A static method called *encrypt()*, which does the following:
    a. Follows the algorithm outlined in the previous page to encode a message given a `String` and `int` passed in as parameters.
    b. Returns the encoded message.

3. A static method named *userInteraction()*, which does the following:
    a. Creates and uses a Scanner and its *nextLine()* and *nextInt()* methods to read input from the user to get values for the message and key.
    b. Further tests the encrypt method using these values.

Advice: Build this method in parts, compiling and running after each major addition to your code to check that the code does work as you expect. For example, you might first create the String and int variables in the *testing()* method and then work on the *encrypt()* method and initially test its implementation with simple values (don't worry about wrapping around Z right away). After that, write further tests in your *testing()* method and figure out the complicated cases in your *encrypt()* method.

File to submit: Encode.java

Marking

Your mark will be based on the following criteria:

- Your code *must compile and run*. It must prompt the user, read text input, and produce the expected output as described and shown above.
- Your code must conform to the requirements mentioned above (i.e., have *testing(), encrypt()* and *userInteraction()* methods.
- Your code must follow the guidelines outlined in Style_Guidelines.pdf, found through the Lectures & Stuff link in the Lab Resources folder on connex.