Double-Caesar Cipher

1 Overview

An important class of algorithms that some computer scientists concern themselves with are encryption and decryption techniques. How can text be encoded so that no one can read it apart from the desired recipient? This is critical for trade secrets and government messages, but even more critical for such common needs as secure web-based purchases, being able to send your password over the web without others listening in, and so on.

An old simplistic trick is the Caesar cipher. Pick a key from 1 and 25; then for each character in your message, shift each letter forward by the key, wrapping around the end of the alphabet. For example, if your original message is `helloyou`, and your key is 2, your encrypted message is `jgnnqaqw`. Supposedly, Julius Caesar used this technique to communicate with his generals. It is very easy to crack, however.

A slightly harder but dramatically more effective variation is the double-Caesar cipher. It works just like the Caesar cipher, but each letter in your message gets shifted by a different amount. How much? It is based on another string of text, called the key. Each letter in the key tells you how many letters to advance: an `a` is 0, a `b` is 1, and so on. For example, if your original message is `helloyou` and your key is `dog`, the `d` in `dog` means that you shift the first letter of `helloyou` 3 letters, the `o` in `dog` means you shift the second letter of `helloyou` by 14 letters, and the `g` in `dog` means you shift the third letter of `helloyou` by 6 letters. You then repeat the pattern: the fourth letter of `helloyou` gets shifted by 3 letters, the fifth letter gets shifted by 14 letters, and so on. This produces the encrupted message `ksroceri`. Double-Caesar ciphers are actually quite hard to crack.

Just to try to help make it clear, here's that same encoding again:

```
Original:      h e l l o y o u
Repeated key: d o g d o g d o
Encrypted:    k s r o c e r i
```

2 Your assignment

You will actually write two programs.

2.1 The first program

The first program, called `Encoder.java`, should have the user first enter in a message in English, and then a key. Your program should assume that your input text is entirely in lower case, with no punctuation, and no spaces. `Encoder.java` should read in this information, then print out to the screen the encrypted message via the double-Caesar cipher.

Here is a sample run below.

```
What is the original message? helloyou
What is the key? dog
The encrypted message is ksroceri
```

2.2 The second program

Your second program, called `Decoder.java`, should read an encrypted message and a key. Your program should assume that your encrypted text is entirely in lower case, with no punctuation, and no spaces. Similarly to the above, your program should first prompt for the encrypted message, and then prompt for the key. Your program should print out the decoded message to the screen.

Here is a sample run below.

```
What is the encrypted message? ksroceri
What is the key? dog
The original message is helloyou
```

3 Bonus extra

If you want to push yourself a little harder allow for spaces between words. The catch is that you should also encode the spaces, so that the person looking at the encrypted message can't tell how many letters each word in the original was. To do this, the simplest way is change your setup so that you think of your alphabet as having 27 letters: the original 26 letters and a space. This means that your encoded message may have spaces in it, but they may not correspond to positions where spaces were in the original.

Author: Dave Musicant