CS3 Project 1: Cipher (Spring 2024)

Project description

In this project, we'll write a program that can encipher and decipher text using known offset and block size values, and even crack ciphers with unknown values by trying all reasonable combinations of values while checking the deciphered text against simple punctuation and capitalization conventions. As an example, enciphering the clear text string "Test offset three, block length two." with offset 3 and block size 2 gives the enciphered text string "hWwvr;iihv;wkwhu.he;ronfo;qhwj;kzwAr", and deciphering it with the same offset and block size gives back the original clear text string. Although the operations work on strings, please use the char[] representation of those strings when enciphering and deciphering, so that we can transform the sequence of characters in-place while getting practice using arrays.

To encipher a string, we do two operations in either order: apply the offset to each character and reverse the order of characters in each block (and any partial block left over at the end).

Applying the offset

The offset, which must be between 0 and 59 inclusive, is added when enciphering to the index of each character in a length 60 characters array that starts with the upper-case letters in order, then the lower-case letters in order, then a space, a newline, a comma, a semicolon, a colon, a period, an exclamation point, and a question mark. Remember a newline character in Java is written '\n'. If a character isn't found in the array, use $ as the default replacement. Be careful to keep indexes within array bounds by "wrapping around" from the end of the array to the beginning: subtract the array length as needed or use %. When deciphering, we can subtract the offset from the index of each character in the characters array and wrap around by adding the array length (60) as needed, again using $ as the default replacement if the character is not found in the characters array.

Applying the block size

The block size, which must be greater than 0, is used to reverse the sequence of characters in each block of the array we are enciphering or deciphering, starting from the beginning, as well as any partial block left over at the end. When deciphering, we do exactly the same thing.

Cracking

To "crack", or decipher text without knowing the offset or block size, we can try to decipher it with all reasonable combinations of offset (0 through 59) and block size (1 through the length of the text), and under the assumption that the clear text is one or more English sentences we can calculate a score for each candidate deciphered text and keep the one with the best score. For this project, we'll use a simple scoring strategy based on punctuation and capitalization conventions. It's fine to be creative with scoring, as long as your code can crack the examples used in the starter project test cases (or testin.txt and testout.txt files if you don’t use the starter project). Here's an example strategy:

- If the first character is a capital letter, add 2; if it's a space or punctuation, subtract 2.
- If the last character is a period, exclamation point, or question mark, add 2; if it's a comma, semicolon, or colon, subtract 2; if it's a letter, subtract 1.
- If a middle (not first or last) punctuation character comes after a letter, add 1; if it's after a space, subtract 2.
- If a middle (not first or last) punctuation character comes before a space, add 1; if it's before a letter, subtract 2.
- If a middle (not first or last) character is a space, add 1

Public constructor and methods of the Cipher class (private methods are optional)

- public Cipher(int offset, int blockSize)
- public String encipher(String clear)
- public String decipher(String enciphered)
- public static String crack(String enciphered)
- public static void main(String[] args) // OK to put this in a different class

Interactive behavior

The main method should first print a welcome message then prompt the user in a loop for whether to encipher, decipher, crack, or quit, and then prompt for additional information as needed (offset, blocksize, clear or enciphered text). Please allow users to enter newlines as \n so that input text can be a single line, and then your code can replace them with actual newline characters. Don't convert newlines to \n in the results you print. Once you have called the appropriate method, just print the result. Example (user input in *italics*):
Welcome to Cipher! Please enter all text as a single line, substituting \n for newline characters.
Would you like to encipher, decipher, crack, or quit? *encipher*
Please enter an offset (0-59): *3*
Please enter a block size (>= 1): *2*
Please enter text to encipher: *Test offset three, block length two.*
hWwvr;iihv;wkwhu.he;ronfo;qhwj;kzwAr
Would you like to encipher, decipher, crack, or quit? *quit*

Tests

The main method of the Tests.java file has 16 tests for the Cipher class, 2 of which pass in the starting project. There's a placeholder string for the expected value of the last (crack) test, so you'll need to figure that out and replace that value to make the test pass. You may also add additional tests. If you choose not to use the starter project, you can find the same test inputs and outputs in the testin.txt and testout.txt files (the outputs include very specific prompts for input).

Collaboration

This project works best in groups of two, where partners sit together in class and work together, and collaboration is strongly encouraged. Please help each other design, test, and debug project code. You will still turn in your own reflections and code. It's OK for partners to turn in the same code, although personalizing your code is encouraged especially if you're adding extra features.

What to turn in (as a link to a Google drive with the following three files):

- Cipher.java: Be sure encipher, decipher, and crack all work interactively.
  - Optional: If you put main in a different class, turn that file in too.
- Tests.java: Fill in the expected value for the last test, make sure all tests pass.
  - Or, testout.txt with the expected value for the last test filled in
- A brief (one or two paragraph) reflection about your individual experience on this project.

Grading rubric (25 points total)

- Readable code following Java conventions, use of char[], thoughtful reflection: 5 points
- Interactive experience, main method calls encipher, decipher, crack: 5 points
- The encipher method is complete and passes associated tests: 5 points
- The decipher method is complete and passes associated tests: 5 points
- The crack method is complete and passes associated tests: 5 points