1. Name:

Project 5

1. Instructions

Project 5, Substitution Ciphers   
  
Substitution ciphers that encode a message by substituting one character for another go back at least as far as Julius Caesar, who used a rotating character scheme to encode military orders. This simple type of encryption is vulnerable to statistical attacks, however, as anyone who has solved CRYPTOGRAM puzzles can attest. In World War II, the Nazi military employed an encryption scheme that addressed this weakness of simple substitution ciphers. This scheme, implemented by typewriter-sized devices known as Enigma machines, gave the Nazis a tactical advantage that greatly contributed to their early success in the war. In fact, the eventual breaking of this coding scheme by researchers at Bletchley Park, England (including Alan Turing) is hailed as one of the turning points of the war.

Enigma machines used interchangeable rotors that could be placed in different orientations to obtain different substitution patterns. More significantly, the rotors rotated after each character was encoded, changing the substitution pattern and making the code very difficult to break.

The enigma has several rotors, we will assume three for this project. Let's call them left, middle, and right. Each of the rotors are interchangeable. The eight rotors we are working with are shown in the following table:

|  |
| --- |
|  |

For this assignment, you are to design a Java class named Enigma that simulates this three-ring model. The Enigma used in WWII had 8 interchangable rotors, as shown below:

|  |  |
| --- | --- |
| **Rotor #** | **ABCDEFGHIJKLMNOPQRSTUVWXYZ** |
| I | EKMFLGDQVZNTOWYHXUSPAIBRCJ |
| II | AJDKSIRUXBLHWTMCQGZNPYFVOE |
| III | BDFHJLCPRTXVZNYEIWGAKMUSQO |
| IV | ESOVPZJAYQUIRHXLNFTGKDCMWB |
| V | VZBRGITYUPSDNHLXAWMJQOFECK |
| VI | JPGVOUMFYQBENHZRDKASXLICTW |
| VII | NZJHGRCXMYSWBOUFAIVLPEKQDT |
| VIII | FKQHTLXOCBJSPDZRAMEWNIUYGV |
|  |  |

 When a character is to be encrypted it is first applied to the right rotor. Let's assume that Rotor I is in the left position, Rotor II in the middle position, and Rotor 3 in the right position. If we wanted to encode the letter M, we would first use the right Rotor and substitute M with a Z.  We then use the middle rotor to find a substitute for the Z, in this case  E. Then use the left rotor to find a substitute for the E, in this case an L. So the original M gets encoded as an L. Now this is not really that complicated and can be broken with traditional statistical analysis. But what happens next is that the right rotor is shifted one position. If we were to encode another M, the right rotor would now align the M with an N. Then the process continues with the middle and left rotors (N becomes T, then T becomes P).

Whenever the right rotor returns to its original orientation, the middle rotor turns once in lock-step, just like the odometer in a car. When the middle rotor returns to its original orientation, then the left rotor shifts one position.

Note that decrypting a message requires following the same steps, only in reverse (i.e., find the character on the outer rotor, note the character aligned with it on the middle rotor, find that character on the outer rotor, then output the character aligned with it on the inner rotor).   
  
An additional layer of complexity is that each rotor can start at the position shown in the table (the 'A' starting point), or any of the other 25 positions. So if Rotor I started in the C location, an A would be substituted with an M.   
  
Our project is to create a program where

* + the user can select which rotors are used in which of the three positions
  + the user specifies the starting point of each rotor (recall the table shows the 'A' starting points
  + the user specifies whether encryption or decryption is taking place
  + the plaintext to be encrypted (or the ciphertext to be decrypted) is entered
  + the resulting ciphertext (or plaintext) is presented

To get all the necessary information we will be using an input file. The file will contain on the first line the Left Rotor (I, II, III, IV, V, VI, VII, or VIII) and the starting position. The second line with specify the middle rotor (and its starting position), and the third line will specify the right rotor (and its starting position). The text to be encrypted (or decrypted) will begin on the 4th line and 30 characters will be presented on each line until the file ends.   
  
An example file to be encrypted would be   
I A   
II A   
III A   
HELLOTHERECLASSTHISISANEXAMPLE   
OFWHATANINPUTFILEWOULDLOOKLIKE   
THELASTLINENEEDNOTHAVETHIRTYCH   
ARACTERS   
and the corresponding encrypted text would be   
I A   
II A   
III A   
MQPYUXPCOPLHLQFDJOINLBWNBOEKOX   
AYULJUYDQFRHHCFTDIYIHUIUTMUMTI   
VTEGZPDNJQGSJXXLZRJUCJCFZWUJFK   
FBZLGMGY   
  
Need example where rotors don't start at A.   
  
The 55 functional points will be broken down as follows:

* + 5 points displaying project and programmer information
  + 15 points able to set the three rotors and starting positions from the possible 8(I-VII) rotors and 26 starting positions (A-Z)
  + 5 points prompt for input and output filenames
  + 15 points dealing with files appropriately (reading/writing)
  + 5 points prompting for whether to encrypt/decrypt and processing accordingly
  + 10 points encrypting / decrypting correctly (with advancing of rotors).

Looking for up to 10 points of Extra Credit?

* + implement a more accurate Enigma encoding (5 points) portions to implement include
    - reflector
    - actual enigma encoding
    - double steps
  + implement a graphical user interface where the filenames can be entered, a selection mechanism for choosing rotors and starting positions, and a radio button (or the like) for choosing whether to encode or decode.

1. Due Date

April 8, 2012 11:59:00 PM CDT

1. Points Possible

100