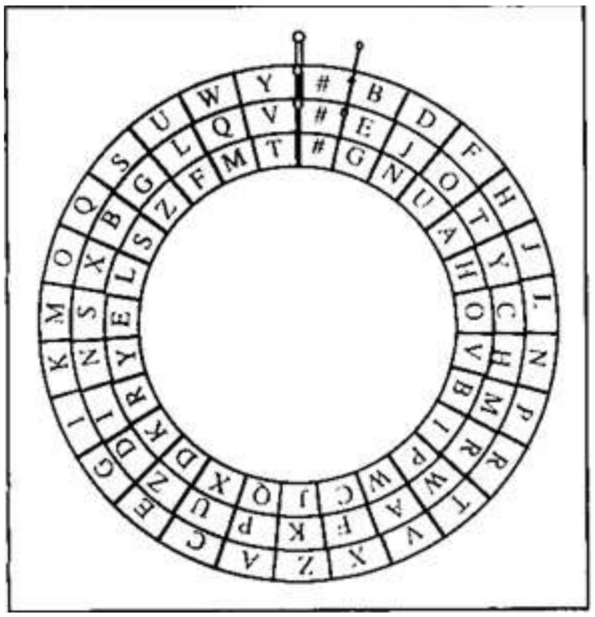
CSC 143 Assignment #1
A Simple Enigma Model[1]

For 25 points
Due Date: Friday, April 24 at 11:59 pm
This is a team project

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 behavior of the rotating rotors can be modeled, in a simplified form, by a device consisting of labeled, concentric rings. For example, the model below has three rings labeled with the letters of the alphabet and '#' (representing a space).



To encrypt a character using this model, find the character on the inner rotor (i.e., the inside ring) and note the character aligned with it on the outer rotor (i.e., the outside ring), then find that character on the middle rotor (i.e., the middle ring) and output the one aligned with it on the outer rotor. After a character is encrypted, turn the inner rotor clockwise one step. Whenever the inner rotor returns to its original orientation, the middle rotor turns once in lock-step, just like the odometer in a car.

For example, in this configuration the character 'A' would be encrypted as 'N', since 'A' on the inner rotor is aligned with 'H' on the outer rotor, and 'H' on the middle rotor is aligned with 'N' on the outer rotor. After performing this encryption, the inner rotor is rotated clockwise, so the letter 'A' would next be encrypted as 'D'.

[1] Adopted from Nifty Assignments - SIGCSE 2009 (http://nifty.stanford.edu/2009/reed-enigma/)

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).

For this assignment, you are to design a Java class named `Enigma` that simulates this three-ring model. You may assume that all Enigma models have the same outer rotor, as shown in the above diagram. That is, the outer rotor consists of the 26 capital letters and the '#' symbol (representing a space) in the following clockwise order: `#BDFHJLNPRTVXZACEGIKMOQSUWY`. Since the other rotors are interchangeable, however, their contents and alignment relative to the outer rotor must be specified when constructing an Enigma model. For example, the initial settings of the inner and middle rotors in the above diagram are `#GNUAHOVBIPWCJQXDKRYELSZFMT` and `#EJOTYCHMRWAFKPUZDINSXBGLQV`, respectively. Using an `Enigma` object, it should be possible to encode and decode text messages, with the appropriate rotation of the rotors occurring after each character encoding/decoding.

You should also design and implement a client program that makes it simple for the user to specify the rotor settings on an Enigma model, and encode or decode text. Once you have your model working, test it by decoding the message `OKNNWRDHGERPILRLAMFZF#FMUC` using the diagram settings.

Note: if you would like to make your own 3-ring Enigma model out of paper, download this online form, cut out the rings, and follow the instructions at the site.

Design and implementation guidelines:

- Javadoc comment all class files and methods
- Use validation and generate and handle exceptions as appropriate
- Structured code - use methods to eliminate redundancy and break large methods into smaller, logical sub problems

Submit the following by using our online class assignment submission:

1. Source-code files (*.java) – one submission per team
2. Sample runs to demonstrate the key features of your program along with the test code that produced this output – clearly mark what output is what
3. Summary of your work – problems/issues with this assignment, any additional things/features that you've thought of etc. Feel free to combine items: 2 and 3 in a single text document
4. Place all of the above files in a folder, zip the folder and attach to submit
5. Also, you need to email me an individual report on your contributions to this homework and what your teammate contributed. Break this down into a nice looking chart of work items, the owner and % of the work (adding up to 100%). You need to email me this using Canvas email.