PROJECT CHECKPOINT

PROJECT NAME:

Ænigma

PROJECT AUTHOR:

Dan Cassidy

PROJECT PURPOSE:

This application will be able to use the Enigma cipher, developed by the Germans in the early 1920's, to encode and decode local messages. The goal is to be able to mimic the machine settings perfectly, including different numbers & types of rotors and a different reflector.

PROGRESS SO FAR:

First off, I should explain that beyond creating a simple functional UI, I have thus far dealt with only the model portion of this application. In this case, that means a class representing a generic Enigma machine, as well as classes representing the various working parts, so far including the rotors, and the plug board. Unfortunately this means that aside from the slightly refined GUI design shown in this document, I have precious little additional graphical documentation to supply. I will cover code progression shortly, but first I wish to expound upon one of the planned features that I plan to include by the time the final report rolls around.

That afore-mentioned feature is the menu. It has not been added yet (obviously, from the screenshot), but I do have a fairly solid idea of the items that are going to be included on it, and how they will work. The first item will be a "Clear" button, which will (upon confirmation) clear the input and output windows. This will be included on the menu bar itself and not in the overflow. The second, also included on the menu bar and not the overflow, will be a "Reset" button, which will (again, upon confirmation) reset the rotors to their starting point for this session.

The next three items will all be in the overflow section of the menu and are as follows: "Settings", "Help", "About". These items, I feel, are mostly self-explanatory, with the addendum that "Settings" is for application-wide settings, not machine settings. (Machine settings will be accessed by tapping on the Rotor Panel, and will be view-only at first, but can be edited if the user chooses to do so.)

And now, on to the code.

Quite frankly, figuring out how to mimic a complex electro-mechanical device in a computer program, while keeping things both relatively easy to understand and modular, has been extremely challenging. Because of my incomplete understanding of the exact process, bugs have been introduced which throw off the output with the resultant effect that the single hardest thing so far has been the debugging process, since descriptions of exactly how the machine works that have the kind of extreme detail I require are few and far between and spread out all over the internet. In fact, as I am typing this, the code for the model (machine) is approximately 95% working, with one outstanding bug that only crops up when a certain set of values are used for the "ring setting" of a rotor.

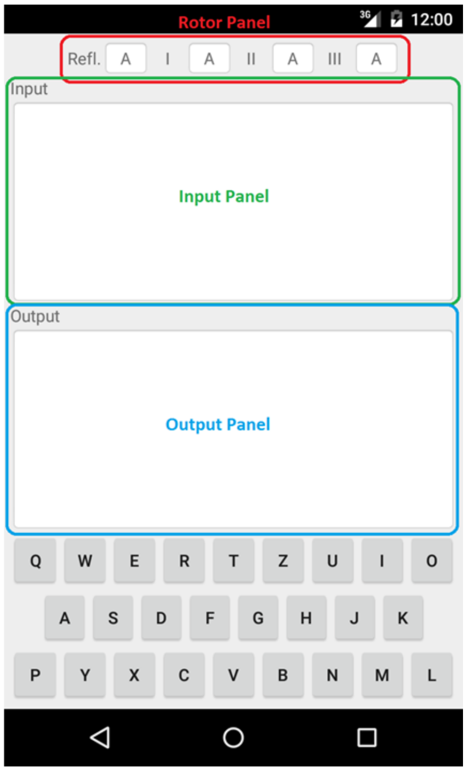
I will include the code files with this report, but will also describe them here in modest detail. I will start from the ground up, so to speak, with the Rotor.java file. This houses the main class that describes all the rotors in an Enigma machine. There are actually three “kinds” of rotors (the stator, the reflector, and the actual rotor) but they function so similarly that they can be generically referred by the same class. The core information about how a rotor functions (for instance, which letters are wired together, whether the rotor moves or not, etc.) is actually stored in the enum file RotorType.java. This file is where all the relevant information about the different rotors are stored, mainly so they can be easily managed.

Next we have the plug board class file Plugboard.java. This is a small class to handle plug board functionality for those Enigma machine types that use it.

And finally, we come to Machine.java. It contains the main class that acts as a gateway to the outside world. I expect that I still have to add some few methods to support altering machine settings on the fly by the settings accessed via the Rotor Panel. This class contains variables of the differing parts, such as rotors and the plug board, and also handles the overall coordination between these parts.

One last note about the java files: they are raw, without any sort of explicit preparation for readability, so they may have incomplete or missing comments, overly-long lines, and so on.

GUI Design v2:


Rotor Panel
12:00
Refl. A I A II A III A
Input
Input Panel
Output
Output Panel
Q W E R T Z U I O
A S D F G H J K
P Y X C V B N M L