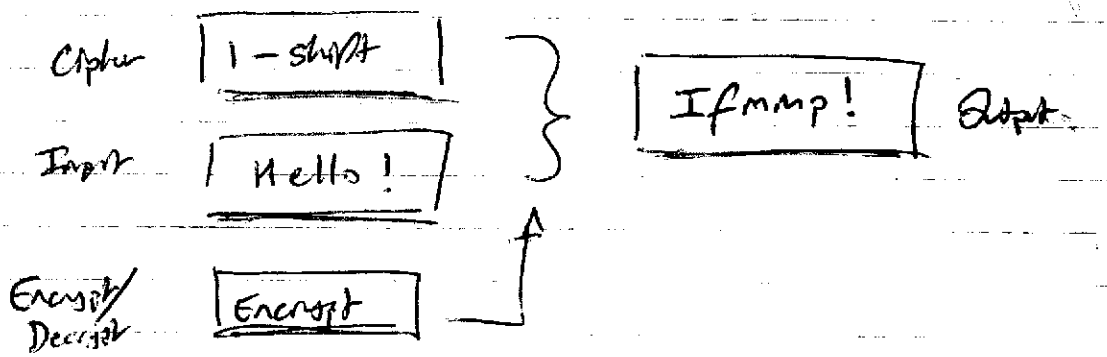
23/7. IDEA FOR "CRYPTOTESTER" PROGRAM

Problem Specification: I want to design a program that allows a user to select a cryptological cipher to ~~encode a typed message~~ encode/decode a typed message and display the output.

Concept:



- The application should be designed in such a way as to make adding crypto algorithms as easy as possible (if possible, no recompilation/reinstallation should take place.) — this fact will likely dictate which programming language we need to use.

We also require a GUI; this will likely be done using Glade for the GTK+ framework together with our required language.
The GUI should be simple for a user to work with (input on the left, output on the right). The user enters their message in a textbox, chooses a cipher from a drop-down menu, and then encrypts OR decrypts it according to the selected cipher. The resulting text appears in the right-hand textbox.

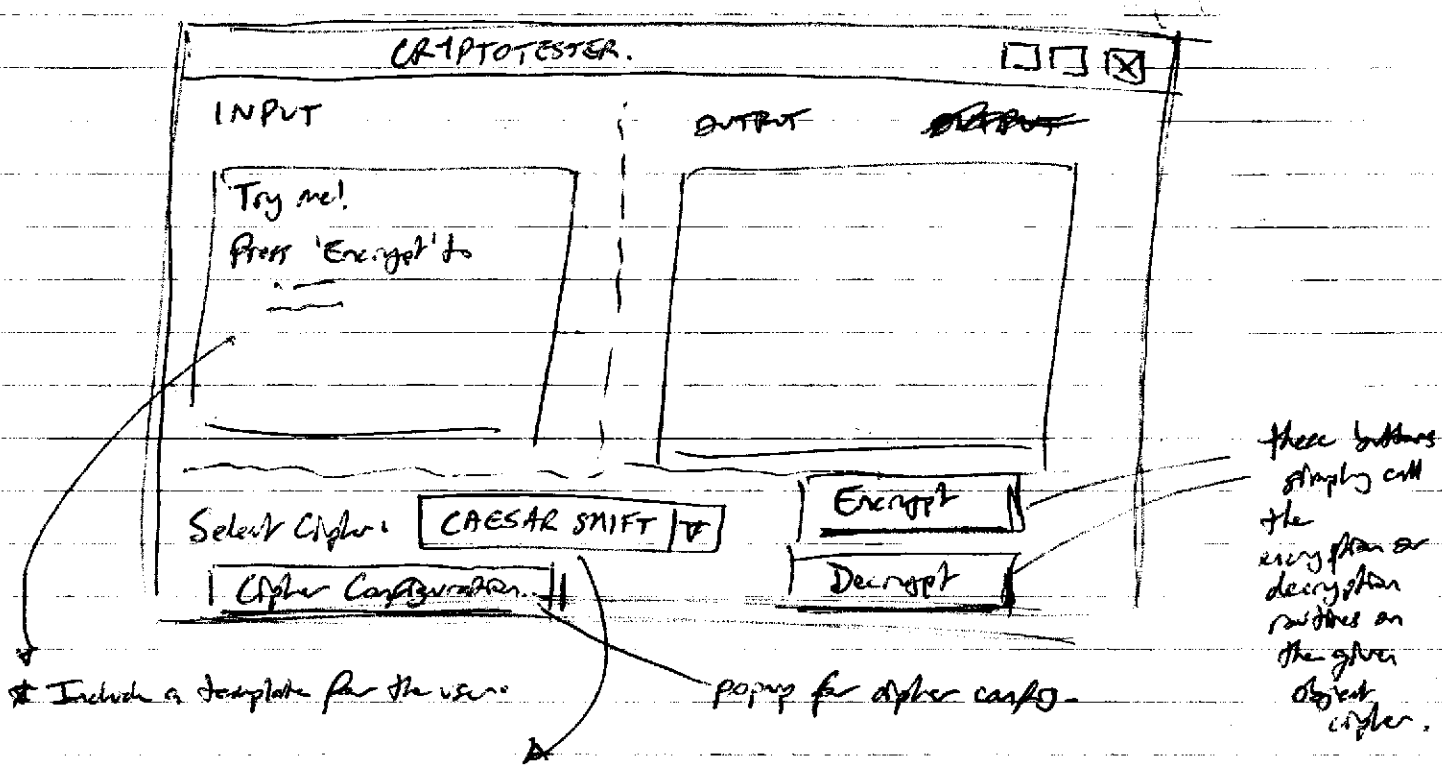
* Some ciphers will require extra parameters. In order to account for differences in setting up these ciphers, we will likely require a popup window to handle cipher configuration, and this window will be dependent on the characteristics of the cipher under consideration.

This fact prevents us from simply being able to add new classes for the ciphers at runtime.

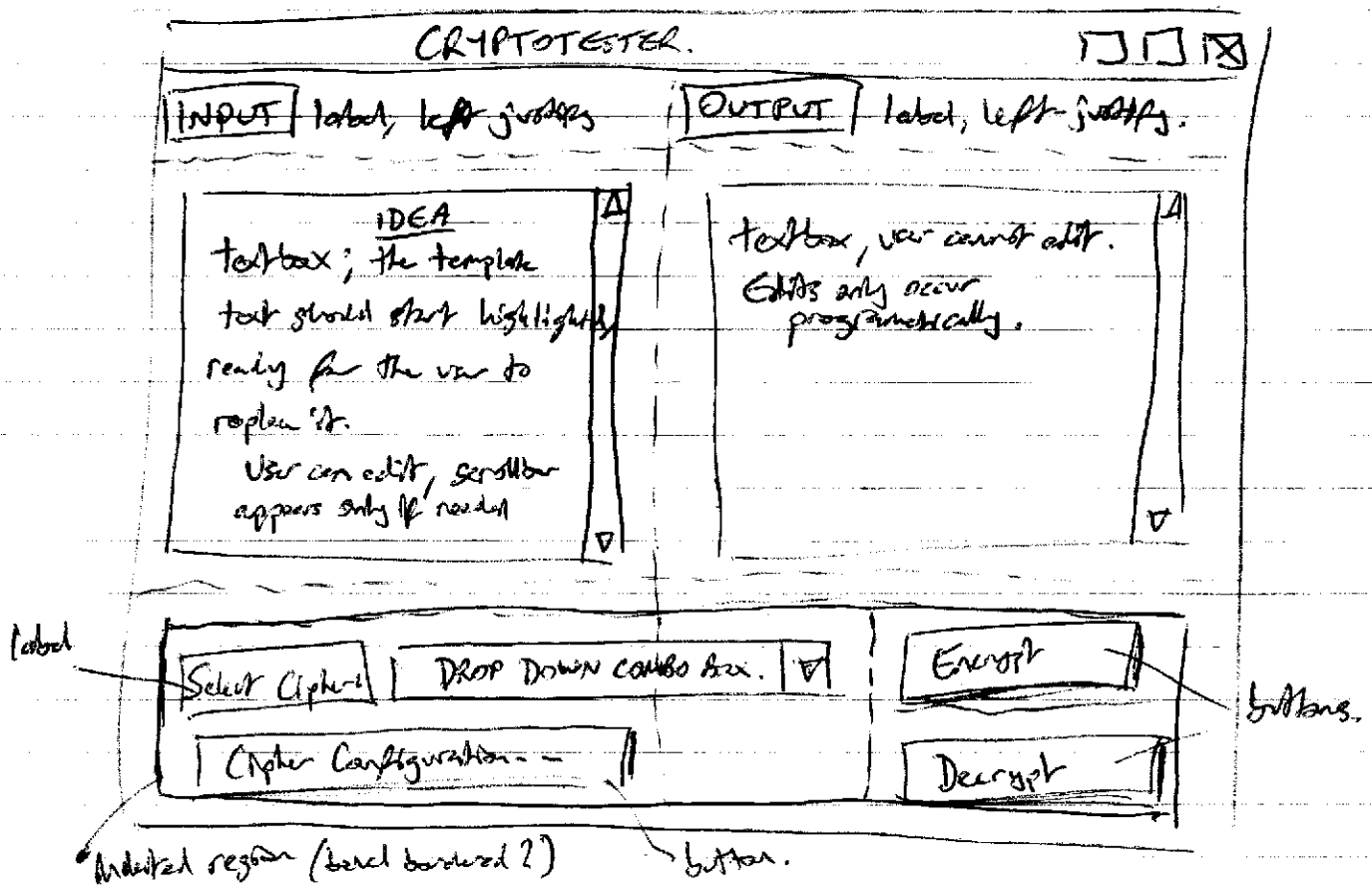
23 7 2014

CRYPTOTESTER SPECIFICATION + DESIGN

GUI CONCEPT:



GUI DETAIL:



24/7

CRYPTOTESTER DESIGN SPEC.

OPTIONS:

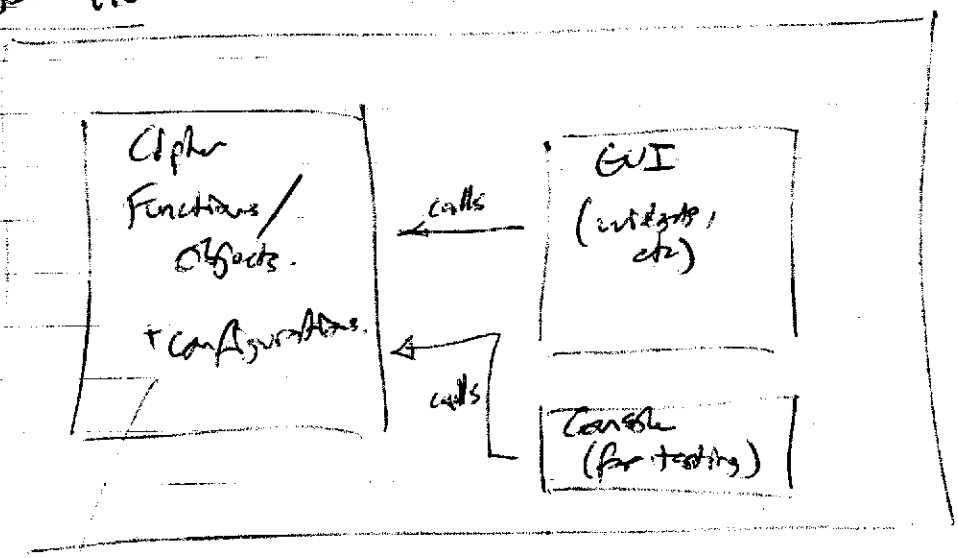
1. Programming languages: Python, Ruby, Scala?
   - Python and Ruby would use the Glade GTK+ framework for the GUI.
   - Scala can use the Swing JVM framework, apparently.

2. Paradigm: probably class-based for the ciphers, but feasibly we could also just use functions in modules (unless this doesn't work with the need for cipher configuration state information...)
   GUI will be event-driven. A minimal amount of multithreading should be necessary — just enough that the GUI doesn't freeze up.
   * Alternatively, encryption/decryption shouldn't take long, so we might not require multithreading at all.

WILL NEED TO EXPLORE PROS/CONS FOR EACH.

✓ Can use Scala to get experience/projects for the functional paradigm (this cryptology tester program might easily be written under the functional paradigm too), instead of necessarily relying on classes.

23/7 : Subsystem View



These ciphers are independent of the GUI and the console — the frontends simply send data to these cipher objects and receive data back.

CRYPTOTESTER DESIGN SPEC.

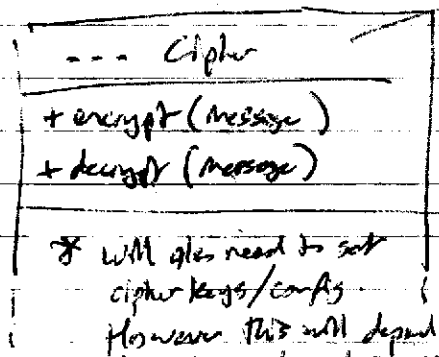
Subsystem Details

Cipher Functions/ Objects.

Each cipher should be isolated in its own file (regardless of which method is used, functions or objects.)
The interface should be common between them — this might suggest that objects are the better solution.

Common Object Interface:

(Alternative to consider — have a main cipher method that takes a function for the cipher as a parameter?)
(i.e. as a closure?)



labelling

CONTINUE DESIGN AFTER NEW DESIGN + SPEC FOR TRANSPORT TABLEAU.

(maybe make changes to spec...)

CIPHERTESTER.

REVISITING CRYPTOTESTER PROGRAM.

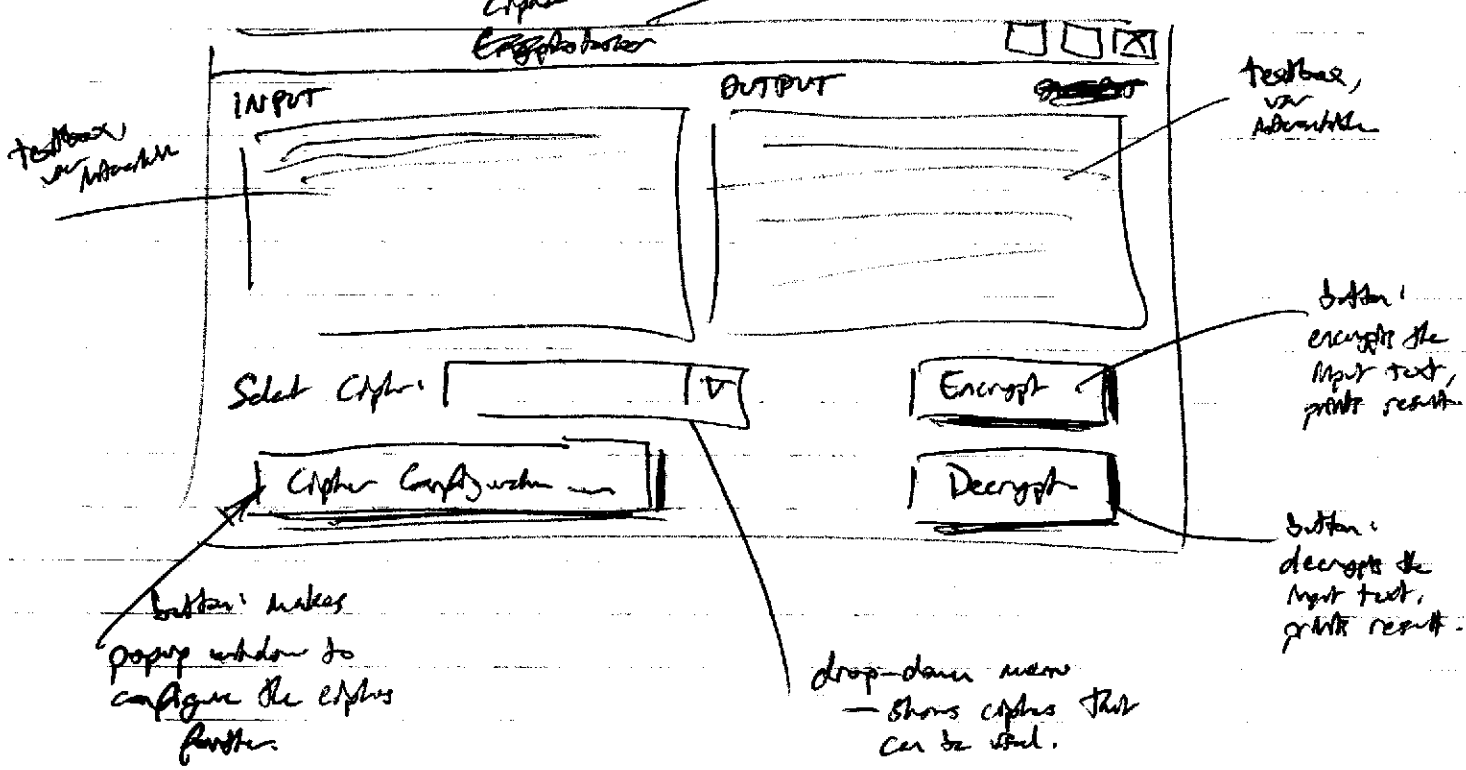
23 11 2014

Specification Overview:

To design a program that allows for the exploration and testing of cryptographic ciphers. The main program will consist of a GUI that allows the user to enter text into an "input" section and spit it out as "output" on the other side after a cryptographic transformation has been applied — the user can either decrypt or encrypt the input text.

To gain experience with using Clojure and its associated tools (leiningen, Swing interop, etc) the program will be done in Clojure, using the Swing GUI framework.
* Will need to finish 'Programming Clojure' so that we know the basics of this?

The GUI will appear as follows: (draft) working name. Change to something else?



* In Clojure, most of the cipher stuff can be done with: change to lower case first? and remove spaces?

(apply str (map #(caller cipher, mod 26 for alphabet) (seq input)).

* PROTOTYPE — to start with, don't worry about multiple ciphers or customisation; just get working on the Ceaser cipher first.

23 11 2014.

CIPHERTESTER PROGRAM.

FUNCTIONAL SPECIFICATION:

- INPUT — Input for the program will mainly come in the form of input strings that are to be encrypted/decrypted.

Alphanumeric / Non-alphanumeric characters.

All characters are allowed (even non-alpha-numeric, though these won't be modified by the cipher.)

- Alternatively, the user might need to configure certain ciphers with keys, etc. before use.

- OUTPUT — once the user has entered some text and clicked 'Encrypt' or 'Decrypt', that operation (determined by the current cipher) is performed and the resulting text appears on screen on the right-hand side.

Only alpha characters are transformed / the rest are echoed.

* The cipher will first convert all the characters of the input to lower-case, and then perform the cipher only on the alphabetical characters! The output will always appear in lower-case.

USER TASKS — 1. Enter input text.
2. Encrypt/Decrypt (Encipher/Decipher?)
3. Modify cipher configuration.
4. Get cipher text (highlight, copy/paste, etc.)

1. User types the input text into a TextField.
2. The Encrypt/Decrypt tasks operate directly on the String containing the text from the user. Once completed, the modified String appears in the output TextField.
   - Encrypt and Decrypt are functions that are kept track of in a variable whose value is based on the chosen cipher in the drop-down box.
   - The ciphers will act on whole Strings and return the result String.
3. Upon modifying the cipher, the information entered by the user is passed to a cipher generator function that sets the *current-cipher* variable to point to a slightly new cipher.

* All operations should be almost instantaneous for most text cases (and obviously taking longer for very large inputs or complicated ciphers.) and ciphers.

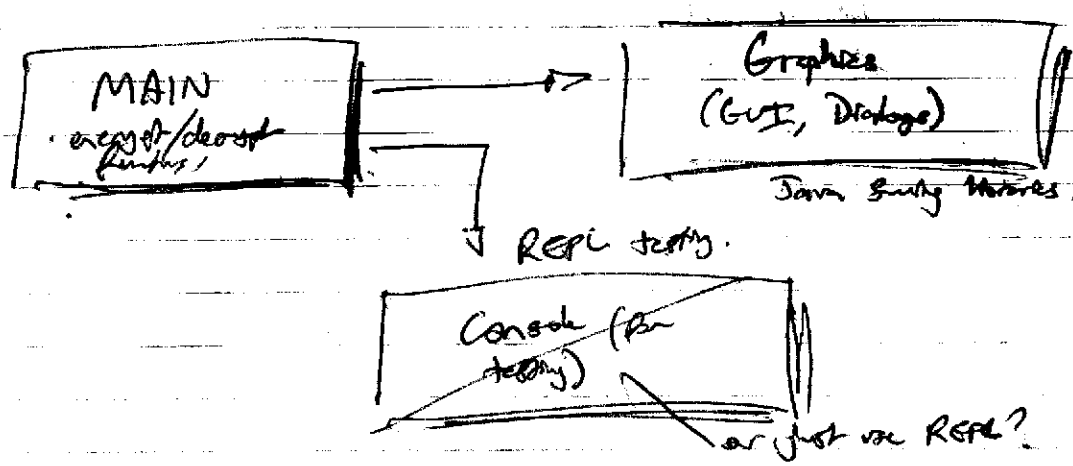
* The program is to have a focus on maintainability, so that new ciphers are very easy to add. (only need to define encrypt/decrypt generator functions that use inputs from the user in a special cipher-configuration dialog.
   - And provide a sensible default!)

24 11 2014

CIPHERTESTER PROGRAM.

- Planning — need to:
  1. Workout program architecture. — MAIN IDEAS: · Function generators for the ciphers.

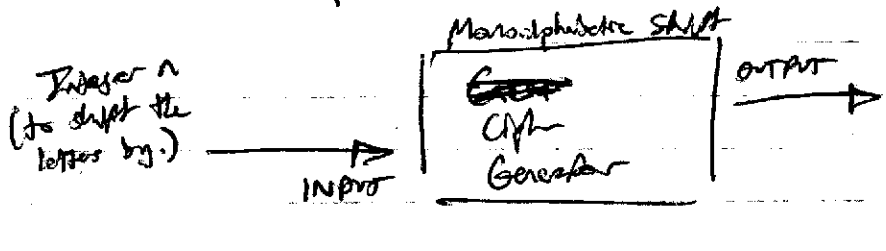
SYSTEM OVERVIEW



Main way the program works:

1. We have a number of Cipher Generators.

* All functions will take a string, return a string. (output)



- this actually seems like it will be the easy part.

2. Need to define a testing strategy for each cipher (so we probably want these namespaced)
   eg. caesar-cipher.clj, caesar-cipher_test.clj.

3. Each cipher will also need a corresponding dialog to show in the GUI when specifying the cipher configuration. (Store these in the -cipher.clj files?

4. The main GUI should be easy to set up — the configuration dialogs should be the only thing that changes (we need a list of the ciphers for the drop-down box — worth looking into reading this in at runtime?)

5. MAIN POINT — should be focus on maintainability and modularity.
   (should only need to modify the cipher list (or this could be dynamically read in) and create new files for the cipher, dialog and test.).

25 11 2014.

CIPHERTESTER PROGRAM.

MONOALPHABETIC SHIFT.

* Can create ancillary functions:

Mod 26 — for the alphabet. (defn mod26 [n] (mod n 26)) } seems about pointless to have this as a separate function.

shift-by-n — go to the nth letter. (defn shift-by-n [character n] (char (+ (int \a) (mod 26 (+ n (- (int character) (int \a)))))))

swap order — can see pattern.

can test this thoroughly, but seems to work fine at the repo.

encipher-generator — returns a function that enciphers. (defn encipher-generator [n] (fn [m-str] (apply str (map (partial shift-by-n) m-str))))

WILL need separate encipher / decipher generators, that take the same keys.

25/11 CREATED CODE REPOSITORY.

- Need to: come up with a good README for the code page. (markdown help?)
- Add test code for the monoalphabetic shift ciphers.
- Add some documentation to the docs/doc folder.

— some links to the other files with (:require monoalp-cipher :as ——)

NEXT STEPS FOR PROJECT (could have this up and running by end of November!)

- Work on + finish GUI mark (ie. w/o configuration yet.)
- Add a README not on github profile — markdown help, ... etc. (tricky?)

need to figure out how this has to work — should it be too different.

* IDEA — use (namespace ...) resolution to differentiate between the different ciphers?

- Add more ciphers: Hill cipher? Vigenere square? RSA cipher? — See "Modern Cryptanalysis".
- custom monoalphabet (user picks mapping | A B C ... X Y Z | / | b c d ... y z a |).

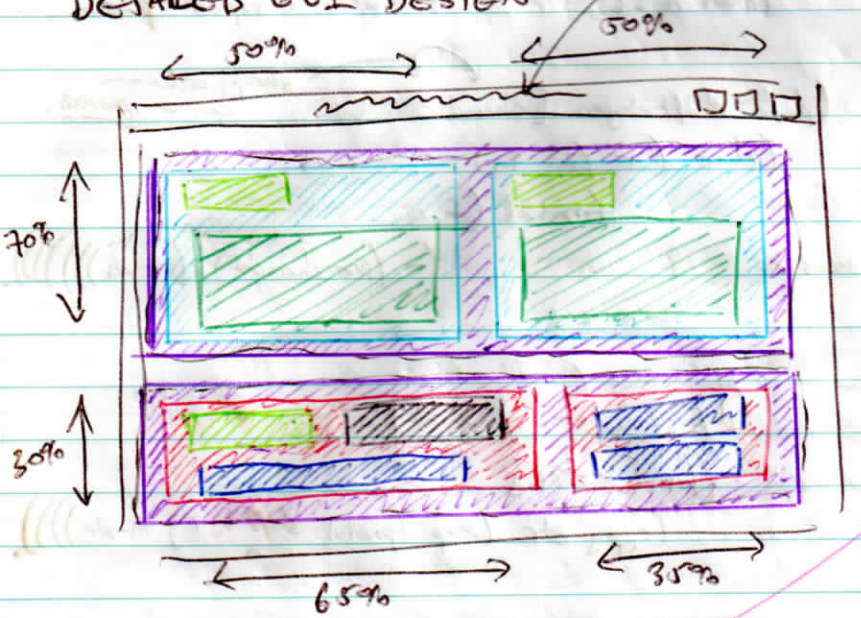
25 / 11 / 2014.

CIPHERTESTER PROGRAM

Working project name: cipher_tester.

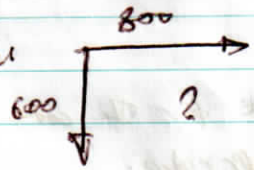
Program name appears in the window title, along with version #.

DETAILED GUI DESIGN



These are also BoxLayouts themselves!

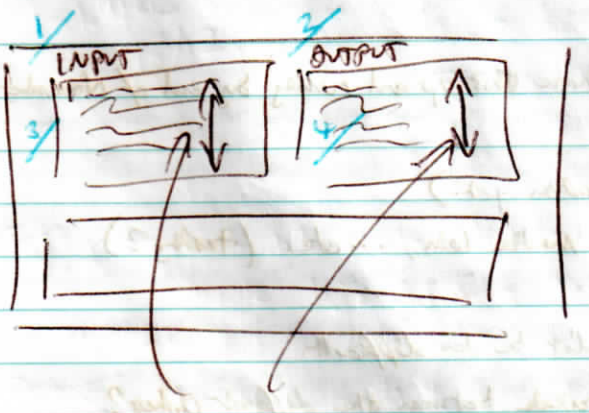
Default window size:



Container Hierarchy:

- JFRAME
  - BOXLAYOUT (VERTICAL)
    - BOXLAYOUT (HORIZONTAL)
      - PANEL
        - LABEL
        - TEXTFIELD
      - PANEL
        - LABEL
        - TEXTFIELD
    - BOXLAYOUT (HORIZONTAL)
      - LABEL
      - COMBOBOX
      - BUTTON
    - BUTTON
    - BUTTON

PROGRAMMATIC VIEW:



Ideally, only the textfields will resize vertically. (Not the Labels or the rest of the other stuff in the frame!)

1, 2/ Labels "Input" and "Output". They do nothing once created, don't need to keep track of them in the program.

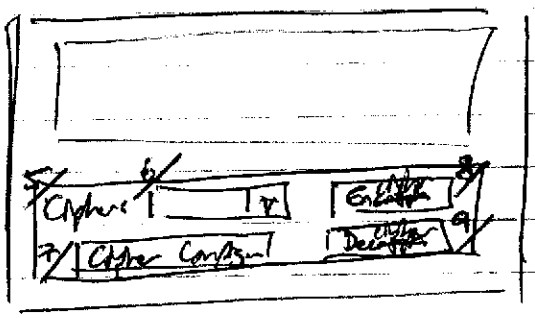
3, 4/ Textfields. User can type in INPUT (maybe there should be a "Try me!" message?). When one of the cipher buttons is clicked, all text in the input window is converted, appears in OUTPUT.

- User can type in the output window, but nothing happens.

25 11 2014.

CIPHERTESTER PROGRAM.

GUI DESIGN — PROGRAMMATIC VIEW.



5/ Label, "Cipher:". Does nothing once created, we don't need to keep track of it.

6/ Drop-down combo box holding names for all of the ciphers currently implemented in the program. (Need to figure out an elegant way to do this.) Selection of the cipher that takes place here, and then it gets customised in a separate dialog. (we do need a sensible default though.)

7/ Button with label "Cipher Configuration...". Once a cipher has been selected from the drop-down menu (default: Caesar cipher, with 3-shift) the user clicks here to bring up a special dialog tailored to deal with the given cipher. The dialog will return with the new ~~cipher function~~ encipher/decipher functions to be used (these might need to be global variables...)

8/ Button, text: "Encipher", or "Decipher".

9/ With a cipher selected, clicking this button simply takes the text from the input window, applies the cipher to it, and prints the result to the output window.

Encipher — applies the encipher function to input → prints to output.

Decipher — applies the decipher function to input → prints to output.

Last things to do in architecture planning:

- Include general interface for the cipher functions (encipher/decipher — generator.)
  (string → [fun] → string.).
- describe the ciphers that will definitely appear at some point ← Hill cipher, RSA, Vigenere square, — AES? etc.
- describe I/O handling + error handling processes.
  (ie. strings from textbox, converted to lowercase, etc.)
- coding conventions — all clojure coding conventions. ✓
- test cases done for all ciphers! ——— tests to be designed in this book-first. ✓

* IDEA: map {:encrypt encryptor :decrypt decryptor} - returned by fn!

30 11 2014

CIPHERTESTER PROGRAM.

TESTING:

UNIT TESTING

- eg. monoalphabetic-shift-cipher:
  - for each $n \in [-26, 26]$:
    - the message "abcde---xyz" should be enciphered properly.
    - the message "abcde---xyz" should be deciphered properly.

(generate this input? (testing framework?) / default. handles capitals / or clojure.test.)

- the message "ABC---XYZ" should be enciphered properly.
- the message "ABC---XYZ" should be deciphered properly.

(ignores other characters.)

- the message "123-89#%&_---" should be enciphered properly.
- " " should be deciphered properly.

#IMPLEMENTED 29/11.

* Can extend this idea for the other ciphers.
  eg. Vigenere cipher tests
  "AA-ABB--8--ZZ-Z"
  (length of key)

* Also, decipher ∘ encipher = identity.
  encipher ∘ decipher

* Validation testing: — enciphering/deciphering is symmetric: encipher ⇄ decipher
  Invariants: length of message doesn't change (for monoalphabetic-shift-cipher at least.)

IMPLEMENTED 13/12.

- for monoalphabetic-shift-cipher:
  enciphering/deciphering + permuting message = permuting message + enciphering/deciphering. (position invariant.)

- Testing — looks into regular Unit testing first, maybe ~~[illegible]~~ test-generative for later, more complicated ciphers.

DONE: 30/11 — Made some tests for the monoalphabetic shift cipher.

1 12 2014.

CIPHERTESTER PROGRAM.

GUI — in core.clj (need -main method?)

Need to import: JFrame —
- JBoxLayout —
  - JLabel
  - JTextField
  - JComboBox
  - JButton.

└

* Can use clojure-contrib ~~Swing-utils add-action-listener~~ deprecated.

* Figured out how to call java -main from leiningen, used to create a Swing app.

FUNCTIONS:

- Specified in cipher_name.clj files:
  - monoalphabetic_shift_cipher.clj
  - vigenere_square_cipher.clj
  - hill_cipher.clj
  - rsa_cipher.clj
  - ...

The main functions in each of these files are as follows:

Keys → encipher-generator → (fn [string]), output string.

INPUT

keys → decipher-generator → (fn [string]), output string.

same keys for encipher/decipher.

* cipher map: keys → cipher → { :encipher (fn) [string] ↦ string
  :decipher (fn) [string] ↦ string. }

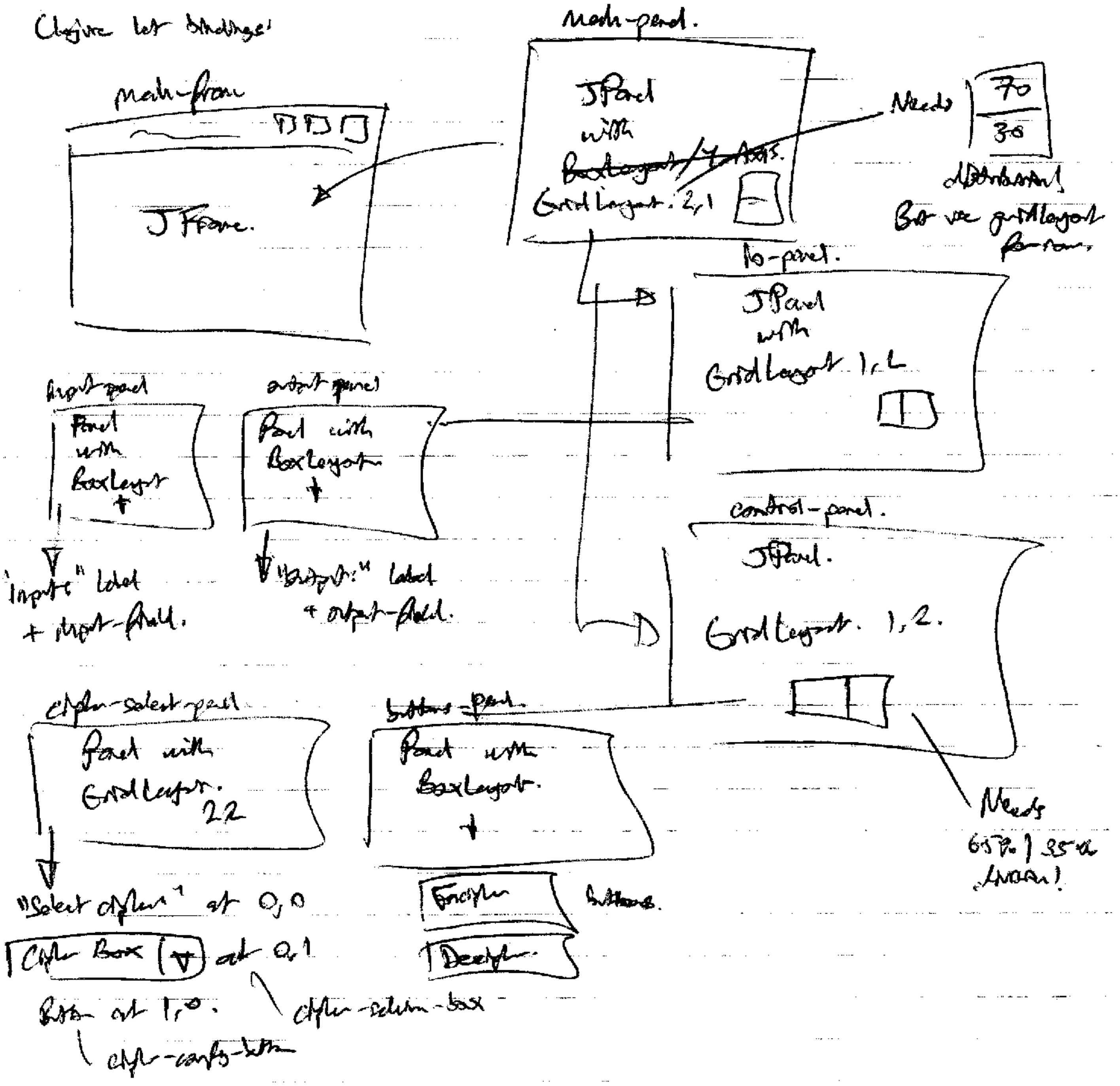
The general idea is to have functions that take in strings and return modified strings.

4 12 2014

CIPHERTESTER PROGRAM.

GUI: Need to specify panels!

Clojure lost bindings!



4/12: Got an initial working for the GUI. Need to add button action listeners, set options for various components (ie. text wrap for input/output boxes, etc.).

14/12: Added action listeners for the buttons.

* For next project, (in Java), find something that allows me to show off my [illegible] knowledge (already shown off plenty of [illegible] with Scala + Clojure)

- 1st release after 'Code Book' ciphers available (Caesar, Vigenere, RSA.)