

# CSSE 371: Lab 2-1

---

## Background

Assume that you are given the world's best text editor library (not! ☺). The text editor allows you to supply it an **InputStream**, which is used to read the input and display the content on the screen. It also allows you to supply an **OutputStream**, which is used to store the content before closing the application window. Here is a typical code that a developer would write to work with this text editor:

```
InputStream in = new FileInputStream("./input_output/in.txt");
OutputStream out = new FileOutputStream("./input_output/out.txt");

TextEditor editor = new TextEditor(in, out);
editor.execute();
```

Please explore and run the application to see how it works. There is, however, an issue with the **TextEditor**. It is not open sourced and is only released as a **jar** library (**Lab2-1/lib/coolest-editor.jar**). Your project is already configured to use this library, so no further configuration is required.

Being a super awesome student that you are, you are thinking, *"Coolest Editor, you cannot stop me from extending your functionality even though you hid your internal code from me! I shall extend thou by adding the encryption facility!"* Here is what you decided to implement as an extension to this library:

**F1.** Ability to store the plain text contents shown on the editor's window to an encrypted output file using the provided simple encryption algorithm (**SubstitutionCipher**).

**F2.** Ability to read from an encrypted input file and decrypt the contents (using **SubstitutionCipher** again) before showing the contents to the user.

As a result, in an ideal world, users with your extension to the text editor can only decipher the contents stored in the encrypted file.

## Design

Create **Lab2-1/docs/Answer.pdf** with answers to the following problems:

**Q1.** Create a UML Class Diagram to present your design idea and explain it in a few lines. [10 points]

## Implementation

**Q2.** Implement your solution in the **Lab2-1/src/problem** package. [F1 - 15 points, F2 - 15 points]

## Testing

**Q3:** Implement necessary test cases in the **Lab2-1/test/problem** package that tests both **F1** and **F2**. [10 points]

[**Note:** The general unit testing convention is that you have at least one test class per concrete implementation class. A test class may have several unit testing methods that check boundary conditions of the methods in the concrete implementation. Example: If you have classes **A** and **B** that inherit interface **I**, then you should create **TestA** and **TestB** class in your test suite.]

## Deliverable

Bundle your project in the **zip** format [**not rar**] and turn it in on Moodle.