Assignment #7: Basic GUIs, Files & Exceptions

*Due: Wednesday, December 13th @ 8:00AM*

*Total Possible Points: 20*

How to Submit

* Submit Visual Studio project in zip or RAR format as a Moodle assignment (no emails or hardcopies accepted)

Goals

* Open and write files using streams
* Use file and save dialog boxes to choose file paths
* Implement a basic Graphics User Interface (GUI)
* Use exception handling to communicate issues to the user
* Write new programs based off of existing code

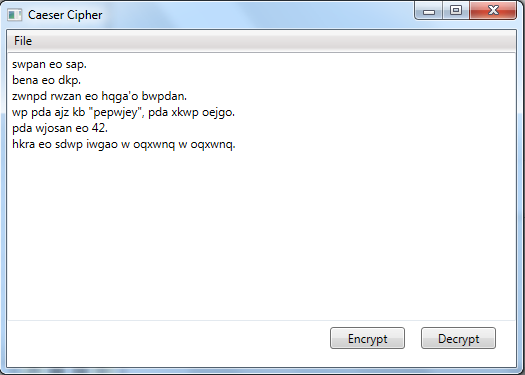
## Your Task

Build a simple WPF GUI application with a text area, buttons, and a menu bar (see the top of the next page) that encrypts and decrypts text files using the Caesar cipher code you wrote in Lab #3.

Your application will allow users to launch file dialog boxes to choose a file to encrypt/decrypt and save. The contents of the file will go into a read-only text area, where users can click buttons to encrypt or decrypt the file, depending on the situation.

You’ll create a separate CeasarCipher class that’s responsible for performing the encryption and decryption. You can use the code you wrote in Lab 3 to implement the class’s Encrypt() method; you’ll need to figure out the Decrypt() method yourself.

From a coding point-of-view, the amount of work is less than Assignment 5’s. However, there will be a learning curve with the GUI portion of the application, especially figuring out how to create menu items in the menu bar.



Row Definitions

Finally, I provided a pair of text files in an **Assignment 7 Files** zip file on Moodle that you can use to test your code (you can create your own text file to test encryption):

* Secrets of the universe.txt - to test decryption
* I am empty.txt – to test your exception handling code

## Requirements/Grading

1. MainWindow.xaml (**GUI file – 6 points total**)
   * Change the title of the Window to “Caesar Cipher” (*0.5 point*)
   * Window’s Grid (*1 point*)
     + Set Height property to Auto
     + Divide into 3 rows
       - Top Row should have a Height of 22 pixels
       - Bottom Row should have a Height of 50 pixels
       - Middle Row should have a Height of 275\* pixels (flexible height)
   * DockPanel (*0.5 point*)
     + Place in top row of Grid
     + Expand to fill Grid
   * Menu Bar (*2.5 points*)
     + **TIP**: Here’s a helpful video tutorial from ElevateCode on setting up a Menu bar in WPF: <https://www.youtube.com/watch?v=IOsb5gHxO2k>
     + Place Menu Bar inside of the DockPanel
       - Set Dock property to Top
       - Set VerticalAlignment property to Top
     + Add a **File** menu (right-click on menu bar > **Add MenuItem**)
     + Add the following MenuItem components to the File menu (right-click on File menu > **Add MenuItem**)
       - **Open File…** menu
         * Name: OpenFileMenu
         * Header: Open File…
         * Create a Click event handler
         * Height: 22
       - **Save File…** menu
         * Name: SaveFileMenu
         * Header: Save File…
         * Create a Click event hander
         * Height: 22
       - A menu Separator
         * Height: 1
       - **Quit** menu
         * Name: QuitFileMenu
         * Header: Quit
         * Create a Click event handler
         * Height: 22
   * Text Box (*0.5 point*)
     + Fills the entire middle Grid row
     + Set its Name to textBox
     + Set IsReadOnly property to true (checked)
     + Set VerticalScrollbarPolicy property to Auto
   * Encrypt and Decrypt Buttons (*1 point*)
     + Place in bottom Grid row
     + Set Names to EncryptButton and DecryptButton, respectively
     + Add event handlers to their Click event listeners
     + Set IsEnabled properties to false (unchecked)
2. CaesarCipher class (**5.5 points total**)
   * Has two (3) private fields (*1 point*)
     + A string named text to store the text to encrypt/decrypt
     + A constant integer named SHIFT to store the size of the cipher shift
     + A constant string named EMPTY\_FILE\_EXCEPTION\_MSG to store the message text of an empty file exception (see below)
   * Has one (1) public property (*1 point*)
     + Gets and sets the contents of the text field
     + set has additional logic
       - If the Length of the passed-in string value is > 0, pass the value to the text field
       - Else, throw a new ArgumentOutOfRangeException; message text is the EMPTY\_FILE\_EXCEPTION\_MSG constant
   * Has one (1) public constructor (*0.5 point*)
     + Takes one parameter: a string representing the text to encrypt/decrypt
     + Initializes the text field via a property
   * Has two (2) public methods
     + public string Encrypt() (*1.5 points)*
       - Using the code you wrote in Lab #3, encrypt a copy of the contents of the text field; returns the encrypted copy as a string
       - NO WRITING TO CONSOLE!
     + public string Decrypt()(*1.5 points)*
       - Modifying the code you wrote in Lab #3, decrypt a copy of the contents of the text field; returns the decrypted copy as a string
       - AGAIN, NO WRITING TO CONSOLE.
3. MainWindow class (**7 points total**)
   * Add using statements to make these namespaces available to your class: (*0.5 point*)
     + System.IO for file handling
     + Microsoft.Win32 for file dialog boxes
   * Has two (2) private fields (*0.5 point*)
     + A string to store the file name of the user’s chosen file
     + A CaesarCipher to store the object that does the encrypting/decrypting
   * Add a private void EnableButtons(bool enabled) method (*1 point*)
     + Sets the IsEnabled properties of the encryptButton and decryptButton to the value of enabled (true or false)
   * Add code to the empty event handlers *created in the previous section* (**NOTE: I assume you used the given names for the menus and buttons, hence the names of the methods**)
     + openMenuItem\_Click()(*2.5 points*)
       - Declare a string variable to store the file contents
       - Create a new OpenFileDialog box
       - Set the dialog box’s initial directory to My Documents
       - Set the dialog box’s filter to just Text files (\*.txt)
       - If the user clicked OK on the dialog box:
         * Assign the chosen file name to the class’s file name field
         * Clear the textBox
         * Disable the Encrypt and Decrypt buttons
         * Read the contents of the chosen file using a StreamReader

Wrapped in a using statement, to automatically close the stream when finished

Read by line, storing the line in the file contents string. **DON’T FORGET TO ADD THE NEWLINE CHARACTER BACK TO THE LINE!**

* + - * + Do all of the following inside a try-catch{} block

Create a new CaesarCipher object

Assign object to the class’s CaesarCipher field

Pass the file contents string as an argument to the constructor

Put the contents of the file in the textBox

Enable the Encrypt and Decrypt buttons

* + - * + catch block catches ArgumentOutOfRangeException exceptions

Display a MessageBox with the exception’s message

* + - saveMenuItem\_Click()(*1 point*)
      * Create a new SaveFileDialog box
      * Set the dialog box’s initial directory to My Documents
      * Set the dialog box’s filter to just Text files (\*.txt)
      * If the user clicked OK on the dialog box:
        + Assign the chosen file name to the class’s file name field
        + Write the contents of the textBox Text Box to the chosen file using a the static File class’s WriteAllText() method.
    - quitMenuItem\_Click() (*0.5 point*)
      * Close the application using the following statement:

Application.Current.Shutdown(); // closes the app

* + - decryptButton\_Click() (*0.5 point*)
      * Calls the CaesarCipher object Decrypt() method
      * Sets the Text of textBox to the returned string
    - encryptButton\_Click() (*0.5 point*)
      * Calls the CaesarCipher object Encrypt() method
      * Sets the Text of textBox to the returned string

1. Comment the CaesarCipher and MainWindow classes **(1.5 points)**
   * *Classe*s: briefly describe the purpose of the class right above the class declaration.
   * *Public methods and constructors*: describe the purpose of each public method and what it returns as output (if anything.)
   * *Major blocks of codes*: briefly describe the purpose & logic of any if(), for(), or foreach() blocks.