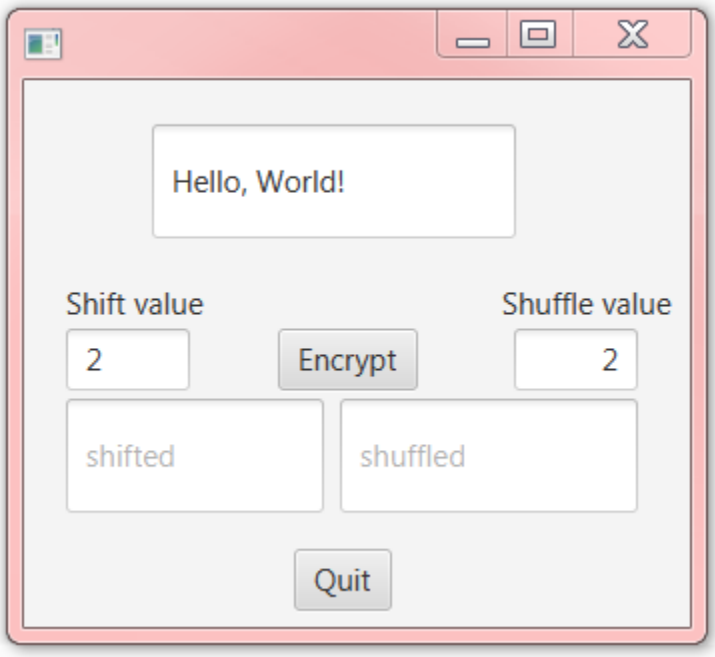
Inheritance and Polymorphism



A cipher is an algorithm for encryption or decryption. To encode is to convert information into a cipher. There are many different techniques for this encoding algorithm. You will implement two of these ciphers. You will also need to be familiar with the String class methods(attached) and Unicode characters(attached).

- Create a JavaFX FXML application using Scene Builder. The GUI should look like the one shown above. The GUI will contain a text field to accept plain text, and text fields to show the encoded text. At the very least, provide a way to trigger the encoding and a Quit/Exit button. Additionally, the GUI must provide input fields to allow the user to enter the shift and shuffle values.
- Create a JavaFX Application using NetBeans or IntelliJ. Define an interface **Cipher** that has a single abstract method **encode**(String plainText), where plainText is the message to be encoded. The method will return the encoded message.
- Define **two classes** which will implement this interface. Both should accept lower case, upper case, or a mixture of both cases.
- Create a class **ShiftN_Cipher** that implements the interface **Cipher**. The class contains one integer instance variable, shifts, which indicates the number of positions to shift the message. Define a constructor with one parameter called shifts to initialize the instance variable. Define a default constructor that takes no parameters and sets the instance data to a random value (range defined by you). Define a private method that shifts a single character using the algorithm described below.
  The encoding of letters should wrap around the alphabet. For example, if shifts is 3, a is replaced by d, B is replaced by E, Z is replaced by C, and so on. A shift value may also be a negative value so that with a shift value of -2, M is replaced by K, Z is replaced by X, a is replaced by y, and B is replaced by Z. Non-letters such as punctuation marks are shifted differently. Start by shifting +161 to the Unicode value of the character. Then, apply the standard shift adjusting it by shifts. You may assume that the encoding will not be higher than the largest Unicode value.
  Define the method **encode** so that each letter is shifted by the value in shift. Method encode should use the private method above. Be sure your method works for both odd- and even-length strings.
- Create a class **ShuffleN_Cipher** that implements the interface **Cipher** . The class contains one integer instance variable, shuffles, which indicates the number of shuffles to be applied to the message. Define a constructor with one parameter called shuffles to initialize the instance variable. Define a default constructor that takes no parameters and sets the instance data to a random value (range defined by you). Define a private method that performs one shuffle using the algorithm described below.
  To perform one shuffle, split the message in half and then take characters from each half alternately. For example, if the message is ABCDEFGHI, the halves are ABCDE and FGHI. The shuffled message is AFBGCHDIE. If shuffles is 2, the second shuffle halves are AFBGC and HDIE giving the final shuffled message of AHFDBIGEC. Define the method encode so that the message is shuffled shuffles times. Method encode should use the private method above. Be sure your method works for both odd- and even-length strings.
- Document each class and each method using Javadoc.

Coding pitfalls to avoid:

Do not enumerate A, B, C, through Z anywhere in your code! The same with non-letters.

Also, add corresponding GUI elements to allow the user to use these features.

- Add a decode method to ShiftN_Cipher which take an encoded message and convert it to plain text.
- Add a decode method to ShuffleN_Cipher which take an encoded message and convert it to plain text.
- Add a third cipher which works as follows.
  Using the sum of all the Unicode values of each given letter (A,E,F,F,J,Y,R) set the seed of a Random object. Using this Random object, create a table of substitutions which map each letter - lower and upper case for a total of 52 entries - to another letter. The mapping is done randomly so that each letter mapping is unique. That is, no two letters may map to the same letter. For example, A maps to K, B maps to W, M maps to D, etc. These mappings are created and stored as part of the object.
  The entries in this substitution table are applied to each letter of the plain text in its encode method. Similarly, the decode method uses this table to reverse the process.