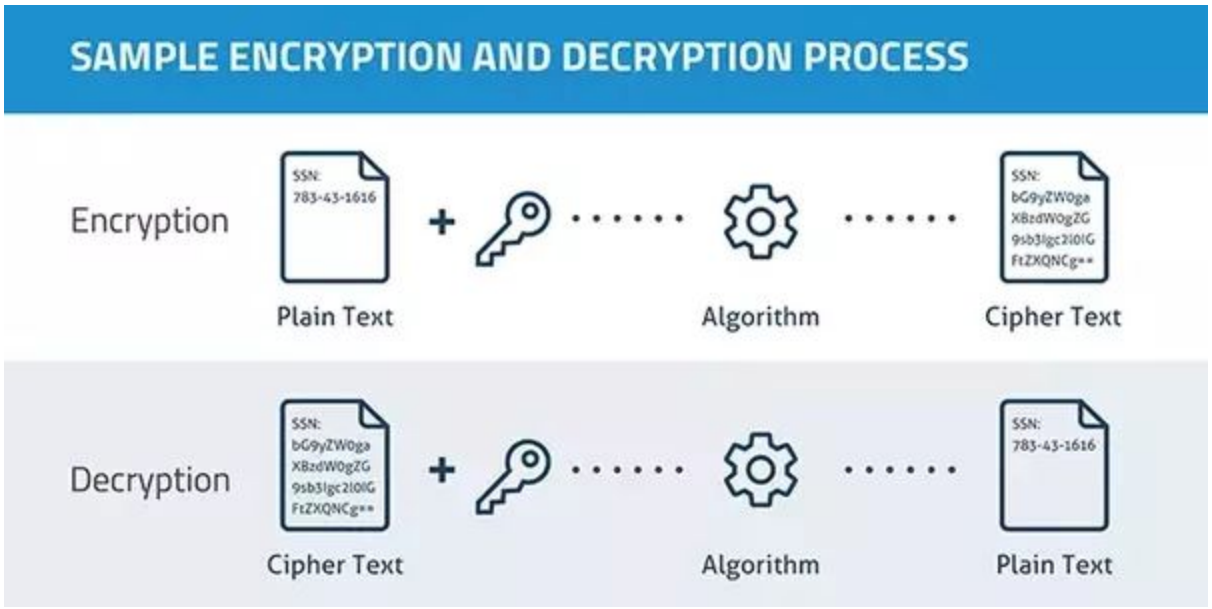
ASSIGNMENT 8-10: INTERFACES

Please review Assignment Guidelines and Academic Integrity on the Course Webpage

A common practice in software is the ability to **encrypt** and **decrypt** information. In this series of exercises, you will be designing a program that can:

1. Encrypt information into a secret code;
2. Check if a secret code is the same value as the original information

If you are unfamiliar with the words encryption, plain text, or secret code (cipher text), refer to the diagram below.



DUE DATE

See course webpage

REQUIRED FILE NAMES

See starter code. You must use the provided starter code as a base for building your solution.

Ensure that you include your course number, student name, and student id at the top of each file.

SUBMISSION INSTRUCTIONS:

1. Zip all files
2. Rename the zip file: **`a810-petersmith-C012345.zip`**
3. Replace **`petersmith`** and **`C012345`** with your student name and id
4. Submit the zip file to assignment dropbox

WHAT YOU NEED TO DO:

You will be building a single application. The application is divided into 3 parts. Complete the requirements for each part and submit all your code in a **single** zip file.

Part 1 - Assignment 8 (2.5%)

- Create the **`IEncryptable`** interface per the description here: Assignment 8 Requirements

Part 2 - Assignment 9 (2.5%):

- Implement the **`IEncryptable`** **encrypt**() and **isOriginal**() functions in the **`SMSMessage`** class, per the description here: Assignment 9 Requirements
- Using the provided Main.java file, test that your implementation works.

Part 3 - Assignment 10 (2.5%):

- Implement the **`IEncryptable`** **encrypt**() and **isOriginal**() functions in the **`TwitterTweet`** class, per the description here: Assignment 10 Requirements
- Using the provided Main.java file, test that your implementation works.

CODING STYLE AND GUIDELINES:

1. All class variables must be set to **private**
2. All interfaces must be named I_____. Example: ISecretMessage, IAnimal, ICleanable
3. All methods inside an interface are marked **public**
4. Within a class, use the *this* keyword to access or update a class variable

5. Outside a class, use the class getter and setter methods to access or update a class variable

ASSIGNMENT 8 REQUIREMENTS: CREATING A ENCRYPTION INTERFACE

Create an interface called `IEncryptable`

The interface has two methods:

- `String encrypt():` Method returns an encrypted string
- `boolean isOriginal(String text):`

Any class that implements the `IEncryptable` interface must provide the algorithm used to encrypt the string, or detect if the provided value is the same as the original text.

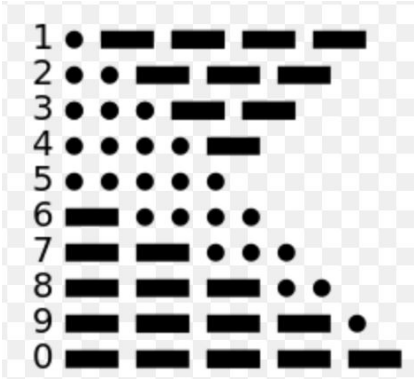
Marking Rubric:

- Interface designed correctly: 1 mark
- Interface provides required methods: 2 marks

ASSIGNMENT 9 REQUIREMENTS: ENCRYPTING A PHONE NUMBER

Create a class called SMSMessage. This class should represent a SMS message that gets sent from a mobile phone.

In this app, the **sender's phone number** can be encrypted using Morse Code. Morse code is a popular cipher where each number (0-9) can be represented using a combination of dot symbols (.) and dash symbols (-).



Example of Morse Code

Starting with SMSMessage.java, implement the methods to allow SMSMessage to conform to the `IEncryptable` interface.

Interfaces:

- This class must implement the `IEncryptable` interface.

Methods:

- **encrypt()**: Note that this function is from the `IEncryptable` interface. Your function should encrypts the sender's phone number using Morse code, and returns the encrypted value as a *String*.

  Please use the provided `MorseCodeUtilities.java` function to assist you in converting numbers to Morse Code.

  - Note that all phone numbers are given in XXX-XXX-XXXX format. Your encrypt() function should **only** translate the numbers, not the - symbols.

    - You may want to use the built in String.replace() function to handle the "-" symbols in the phone number. See more information here: https://stackoverflow.com/a/4576372/13615038

- **isOriginal(String text):** Note that this function is from the **`IEncryptable`** interface. Decrypts the provided value and checks the decrypted value is the same as the sender's phone number.
    - For your implementation of the isOriginal() function, you are **not allowed** to use the **`encrypt()`** function. You must 1) decrypt each morse code symbol into a number; and 2) compare the numbers with the **from** phone number. If the values are the same, return true. Otherwise, return false.
    - Please use the provided **`MorseCodeUtilities.java`** function to assist you in decrypting the provided value.
    - You may want to use the built in String.**`replace()`** function to handle the "-" symbols in the phone number. See more information here: https://stackoverflow.com/a/4576372/13615038

Marking Rubric:

- encrypt() function is correctly implemented: 5 marks
- isOriginal() function correctly implemented: 5 marks
- Class works correctly with the provided Main.java file: 2 marks

Expected Output

This example uses a from phone number = 908-505-1234

```
The following message has been sent by
----.--------.....-----.....----..---...--....- to 905-801-5555
       When are we meeting for dinner? Text me back!

Here is an encrypted phone number:
       ----.--------.....-----.....----..---...--....-
Are the phone numbers the same?
       true
```

ASSIGNMENT 10 - A Twitter Tweet

Create a class that represents a Twitter Tweet.

All tweets can be encrypted using the **Equus Algorithm.**

Methods:

- **encrypt**(): Note that this function is from the **`IEncryptable`** interface. This function encrypts the tweet using the **Equus Algorithm** and returns the encrypted text.

- **isOriginal(String):** Note that this function is from the **`IEncryptable`** interface. This function accepts 1 string parameter containing an encrypted text and decrypts it. If the decrypted text is the same as the original tweet, return true. Otherwise, return false.

Marking Rubric:

- encrypt() function is correctly implemented: 5 marks
- isOriginal() function correctly implemented: 5 marks
- Class works correctly with the provided Main.java file: 2 marks

Expected Output

```
Original tweet:
	If you can protest in person you can vote in person
Encrypted tweet is:
	fIeq ouyeq anceq rotestpeq nieq ersonpeq ouyeq anceq oteveq nieq ersonpeq
------------------------------
Checking message 1:
------------------------------
Encrypted message:
	fIeq ouyeq anceq rotestpeq nieq ersonpeq ouyeq anceq oteveq nieq ersonpeq
Is this the same as the original tweet?
	true
------------------------------
Checking message 2:
------------------------------
Encrypted Message:
	Ieq adheq aeq otleq oteq odeq odayteq utbeq nsteadieq Ieq ookteq aeq apneq
Is this the same as the original tweet?
	false
```

Eqqus Algorithm:

In the Eqqus algorithm, a word is converted to secret text as follows:

1. Remove the first letter of a word
2. Put the letter at the end of the word
3. Append the letters “eq” to the end of the word

Encryption Example

English: I AM SLEEPING
Secret Text: IEQ MAEQ LEEPINGSEQ

Decryption Example:

Similarly, to decrypt the word:

1. Remove the EQ from the end of the word
2. Remove the last letter
3. Put the letter the front of the word

English: PPLEAEQ ANANABEQ
Secret Text: APPLE BANANA