

# Brandeis University COSI 12B, Spring 2017 PA3

Due date: Sunday, Feb 12, 2017, at 11:55pm

## Learning objectives

1. Dealing with String objects and using the StringBuilder class.
2. Defining new classes and methods.
3. Studying about the Set data structure and using the Java Set interface.
4. Reading from a resource text file and initializing your objects with the data from it.

## Background

Cryptography is the study and practice of secure communications, codes and code breaking. From Wikipedia: "Cryptography prior to the modern age was effectively synonymous with encryption, the conversion of information from a readable state to apparent nonsense.

The originator of an encrypted message (Alice) shared the decoding technique needed to recover the original information only with intended recipients (Bob), thereby precluding unwanted persons (Eve) from doing the same. The cryptography literature often uses Alice ("A") for the sender, Bob ("B") for the intended recipient, and Eve ("eavesdropper") for the adversary."

It is an absolutely fascinating area of study, mathematically, algorithmically, but also historically and socially.

This assignment explores some of the simplest approaches to cryptography and one of the earliest techniques. In fact the technique we will look at is old enough that it easily might have been used in biblical times. It is very insecure and very easy to break. Therefore it is a nice basis for a programming assignment. We are looking at what is sometimes referred to as the "Caesar Cipher". We have no evidence that it was used by Caesar, but that's just what it's called.

## Caesar Cipher Technique

In general we describe the class of algorithms used for encoding and decoding messages "Ciphers". There are numerous cipher algorithms, some extraordinarily complex and mathematically sophisticated.

The Caesar Cipher is super simple. It simply considers each letter of the message, and substitutes it with another letter that is "n" positions forward (for negative n, backward) in the alphabet. If  $n \geq 26$ , then it wraps around, so that  $n = 27$  is the same as  $n = 1$ . If  $n = 0$  (or,  $n = k * 26$ , where  $k = 0, 1, 2, \dots$ ), then the ciphertext == plaintext, not a very good way to keep your message secret.

# Encryption

- Given a String *s*, we distinguish between 3 types of characters in it:
  1. Alphabetical character - letters 'a' to 'z' and 'A' to 'Z'.
  2. Numerical characters - digits '0' to '9'.
  3. Other characters - everything that is not alphabetical or numerical character.
- In this assignment, we will encrypt and decrypt alphabetical and numerical characters only. Other characters will not be changed and will be passed through the cipher as they are.
- Encrypted alphabetical character [char] with encryption parameter [N], equals to a new alphabetical character, which is shifted N letters forwards (for a positive N) or backwards (for a negative N) in the sorted English alphabet. If a letter is shifted past the end of the alphabet (after 'Z' or before 'A'), it wraps around. Therefore, a letter 'z' shifted with parameter N=1, equals 'a', and 'a' shifted with parameter N=-1, equals 'z'.
- The encryption is case sensitive, so, uppercase alphabetical character will be encrypted to uppercase, and lowercase to lowercase.
- Numerical characters will be encrypted in a similar way. Encrypting the digit *d* with parameter *N* is equivalent to:  $(d + N) \bmod 10$ . For example,  $\text{Encrypt}(5, 2) = 7$ ,  $\text{Encrypt}(9, 1) = 0$  and  $\text{Encrypt}(0, -1) = 9$ .

## Decryption

Decryption of the encrypted text is defined similarly. The difference is in the direction of shifting. If parameter *N* meant shifting right *N* places for the encryption, it will mean shifting left *N* places for the decryption.

The next equation always holds:  **$\text{Decrypt}(\text{Encrypt}(\text{String}, N), N) = \text{String}$**

## Code-breaking

Even with this very simple approach, an eavesdropper would have to know, or figure out "N" in order to break the code. Obviously if we know the message is likely in English, we could try many possible "N"s and look to see if the message is English. For this assignment you should use the dictionary [Google's 10000 English](#).

## Your Task

Define at least these public methods inside the **CaesarCipher** class:

```
encode(String, int) -> String
decode(String, int) -> String
decode(String)      -> String
```

For clarity you might want to have additional non-public methods.

- `encode(String s, int n) -> String`: Perform a Caesar cipher encryption of the input string *s*, using *n* for the shift. For example:

```
encode("PA3 is the best!", 1) = "QB4 jt uif cftu!"
```

- `decode(String s, int n) -> String`: Perform a Caesar decoding of an input string `s`, given that the shift parameter used for the encryption was `n`. For example:

```
decode("QB4 jt uif cftu!", 1) = "PA3 is the best!"
```

- `decode(String) -> String`: Perform a Caesar decoding without knowing 'n'.
  - **For this method only, you should assume that the message contains only alphabetical characters and spaces to separate the words.**
  - The message will be in English and will contain at least 5 words.
  - There might be some uncommon words, that don't appear in the 10,000 words dictionary. Therefore, if after decoding, you find some words that don't exist in your dictionary, the decoding might still be correct.
  - We will consider a decoded text to be in English, if at least 90% of the words in it appear in the dictionary. (For more info, see guidelines and hints).
  - If, you can't find a corresponding shifting parameter `N`, such that the decoded string is in English, you should return null.

## Guidelines and Hints

- The first 2 methods are quite straightforward.
  - If null was provided instead of a valid String, you should also output a null value. (This is relevant for all 3 methods).
  - You should use the `StringBuilder` class to append an encrypted character to your final output.
- For the third method, you should use the `Set` data structure to store the whole dictionary.
  - First, download the [Google's 10000 English](#) dictionary as a text file (on GitHub page, click Raw -> Save as) and put it under the [dictionary] folder in your project.
  - In your [CaesarCipher] class, create a `Set` data structure and populate it with all the words from the dictionary file. This may be done inside the constructor of your class which may be declared as throwing `FileNotFoundException`. You will be able to access your text file with the following code:

```
File dict = new File("dictionary/google10000.txt");
```

- For every possible `N`, decode the string and check how many words in the decoded string appear

in your dictionary. What is the range of N that is sufficient to check?

- If at least 90% of the decoded words appear in the dictionary, you may consider it as a valid English text and return it as a result. Notice, that the words in the dictionary are all lowercase.
  - If 90% of the words is not a whole value, you should round it down to the nearest whole value. For example, if the string contained 5 words, then 90% out of 5 words is 4.5 words. We round this value down to the nearest whole value -> 4. Therefore, if at least 4 out of 5 decoded words appear in the dictionary, you may return this decoded string as an output. For this method, you may assume that the input string, if it is not null, will contain at least 5 words.
  - `String.split(delimiters)` is a useful builtin method of the `String` class in Java. You may use it to split the given string on spaces by: `String.split(" ")` which will return an array of words in the string.
- Set data structure
    - A Set is a collection that cannot contain duplicate elements. This is why we can store our dictionary in a Set. Java has several implementations for the Set data structure. One of the efficient implementations is using a hash table, which is why they call it a `HashSet`.
    - Once your Set is initialized and contains all the words from the dictionary, you can benefit from a convenient and fast methods to operate on it. For example, you can easily check if a certain word exists in the dictionary.
  - Creating an empty Set of Strings:

```
Set<String> dict = new HashSet<String>();
```

- Adding a new String to dict

```
dict.add("word");
```

- Check if the set contains a word


```
dict.contains("word")
```

- You can study more about Set interface here: [Oracle tutorials](#)
- Tutorial about the [HashSet class](#)

## Extra Challenges

A slightly better version of Caesar cipher is called Multi-Caesar (or Changing Caesar). Add a separate file under `src/main/java` folder and call it **MultiCaesarCipher.java**. Add a public **MulitiCaesarCipher** class with 2 public methods:

```
multiEncode(String s, int[] args) -> String  
multiDecode(String s, int[] args) -> String
```



These 2 functions are similar to encode and decode functions we saw before. The difference is in the number of shifting parameters. Both, multiEncode and multiDecode will receive an array of integers [args] as an additional parameter. This is an array of shifting parameters which will be applied to encode or decode the string.

If only one parameter was given (args array contains a single number) - n, it is a regular Caesar with parameter n.

If 2 parameters were provided, n and m, then the characters at even indices (0,2,4,...) will be encoded with shift-n, and the characters at odd indices (1,3,5,...) will be encoded with shift-m.

With many parameters, n1 ... nX: the first character, and every Xth character after it will be encoded with shift-n1, the second character and every Xth character after it with shift-n2 and so on.

You may assume that args array will contain at least one element.

## Bonus Question Grading

We are going to have approximately 10 programming assignments which as you know will count towards 55% of your grade. Each PA will be scored on a scale of 0-100. Some of those PAs will have bonus questions. They are extra credit. One correctly solved bonus question will give you 20 additional points towards any of the prior or future assignments. For example if you got a 100 score for all your PAs except of one, for which you received an 80, and you did one bonus question, then your overall score for the PA component will still be 100.

## Deliverable

Your deliverable is the Eclipse project exported as a .zip file.

### Terms and Conditions

Cosi12b Assignments by Sandro Golis. Licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.