Practical 16: Text Encryption

In this paractical you will implement a computer program that will use a *monoalphabetic substitution cipher* in order to encrypt text. This kind of encryption is among the most trivial but still requires time and some specialist knowledge in order to be broken.

Procedure: In order to encrypt a message, each letter (or symbol) of the text in the message is simply substituted by another letter (or symbol) provided by a look-up table (or map). The simplicity of the scheme lies in the fact that a specific letter is always replaced by the same alternative. Knowledge of the most frequent letters/words in the language of the message can therefore be used to decode the message.

For example, here is how the sentence:

```
The IPCC's measured assessment shows that the world needs to
face up to the challenge of climate change, and to do so now.
```

might be encrypted using the monoalphabetic substitution cipher with a particular *key* (i.e. look-up table scheme)

```
s5lHjqVV"9H]lM9oflIHM99l99]lGhH95?X9Hh5MhHh5lHX?fYIHGllI9Hh?He
MPlHoDHh?Hh5lHP5MYYlG4lH?eHPYZ]MhlHP5MG4luHMGIHh?HI?H9?HG?X\
```

Note that the key used in the example above may not necessarily be the one that you will end up using in your implementation.

Use a piece of text of your choice for this practical. It could be a paragraph from this practical sheet for example.

Task 1

How would you achieve this task manually (with paper and pencil)? Here is a clue to get you started:

Consider all the printable characters that your keyboard can produce (lowercase/uppercase letters, numbers and symbols). It will take you very little time to type all of them in your program.

Think how you can implement some sort of a look-up table scheme by which each character maps to another. For example:

Character	Maps to
J	O
P	"
a	(
n	y

)	+
;	T
2	*
8	i
...	...
...	...

You may find the following method useful:

```
static void shuffle(String sequence) {
    Random rgen = new Random();

    char[] sequenceAsArray = sequence.toCharArray();

    for(int i=0; i<sequenceAsArray.length; i++) {
        int randomPosition = rgen.nextInt(sequenceAsArray.length);
        char temp = sequenceAsArray[i];
        sequenceAsArray[i] = sequenceAsArray[randomPosition];
        sequenceAsArray[randomPosition] = temp;
    }

    System.out.println(new String(sequenceAsArray));
}
```

Please note that the method above requires that you `import` the `java.util.Random` class.

Task 2

Implement a method that takes a piece of text and a key and returns the encrypted version of the text based on the key.

In order to also test that your method above does the right thing, implement another method that will take an encrypted text and a key and decode the text with that key.

Test your two methods by making use of them in a `main()` method.

Task 3 (optional)

Please note that this can prove to be quite a challenging task. Do not let it distract you from your other study commitments.

Given a sufficiently long piece of encrypted text using the above cipher method, you can have a go at writing code to decipher it.

For this, you will need to analyse the relative frequency of occurrence of letters in the text and use that to compare it with the relative frequency of letters in the English language (assuming you are deciphering an English language text). You can find the relative frequency of letters in the English language on the web, for example at http://en.wikipedia.org/wiki/Letter_frequency.