CC - Cesar's Cryptography

Intent of this exercise.

In this exercise you and your group must take the goal and take it apart in smaller problems.

"problems" can be everything from

- Use an ArrayList here ?
- What is the plan/responsibility of our objects ?
- What is our best plan ? (and what is "best" ?)
- Any usefull Interfaces ?
- And collections ?
- Do we havesome Patterns here ?

Hint: use pen and paper to draw or write them down.

You must use JUnit to :

test – implement – test

the appropriate software to solve the smaller problems.

Collect these results to a complete solution still including the JUnit tests.

Historic Background

Secret messages is an ancient science. And still very actual.

Rulers, their generals and spies has forever had a wish, that their communication should be kept secret, so only the intended recipient could read it.

Caesar, Julius world famous (at least according to himself) general and emperor is the first in Europe, we have some documentation about. He used a letter substitution:

original	a b c d e f g h i j k l m n o p q r s t u v w x y z
coded	d e f g h i j k l m n o p q r s t u v w x y z a b c

That is :

julius is a nice guy	ORIGINAL
mxolxv lv d qlfh jxb	ENCODED

An encoded message of this type is primitive (sorry Cesar), and not very difficult to decode. However, the method is easy to handle and easy to extend. And we could of course include numbers and special characters.

There are 2 important points :

1. every letter correspond to <u>exactly one</u> other letter
2. the "secret" is how many letters we shift

If the coded line above (called the key) is chosen randomly, then it will be much more difficult to decipher for the enemy. And if the key changes often, it is even more difficult.

It is of course a must, that the sender and the receiver know and use the same key.

The weakness in this method ?

We will return to that later.

Remember to use Junit in your development.

CC EXERCISE:

Choose which letters (lower and/or upper case ?), numbers and special characters, thatyou want to include in your crypto algorithm.

Implement a randomizer so you can generate a key as described above.

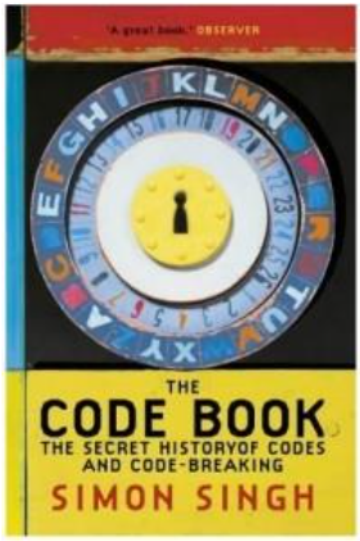
Use this key to encrypt (without looking into the key) Alice In Wonderland or Moby Dick (*.txt files from www.gutenberg.org or on Fronter).

Use this encrypted text (assuming you do not know which it is) to find your randomizers chosen key.

Hint:

Arrays of Char will be fine for the Key, but Java offers something even better. A Collection Class called Maps, which is very good for look up purposes. Chapter 11.3 in Reges.

References :



There is a very good book about Cryptography without all the mathematics :

The Code Book-

Simon Singh

ISBN 1-85702-889-9

And for the real fans of Cryptography Java has a classes for that too (of course) in the package javax.crypto .