Lab 5. Encryption

Purpose
The purpose of this lab is to practice using strings and arrays, methods and parameters.

Problem description
You will write a program that first encrypts then decrypts a message. Here's how a plaintext string is encrypted:

- a 'keyword' is provided e.g. `"elvis"`

- the keyword is used to build an 'alphabet' string, as follows:

  - alphabet begins with the keyword e.g.

    ```
    e l v i s
    ```

  - now the remaining letters 'a' through 'z' are added to the end of alphabet in order. If a letter occurs in keyword, it is not appended to alphabet. (We will work only with lowercase for simplicity) e.g.

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

  - finally, a space is always appended to alphabet e.g.

    ```
    e l v i s a b c d f g h j k m n o p q r t u w x y z ' '
    ```

- if the keyword repeats letters, these are not put into the beginning of alphabet e.g.

  - if keyword is `"elvispresley"`, alphabet will begin

    ```
    e l v i s p r y
    ```

  - then remaining letters are appended in order, followed by space

    ```
    e l v i s p r y a b c d f g h j k m n o q t u w x z ' '
    ```

- now the alphabet is used to convert the 'plaintext' message into an array of integers called 'encrypted', which is the encrypted message

  - look up each plaintext char in the alphabet, and enter the index into the encrypted array

- e.g. (using the `"elvispresley"` alphabet above) if plaintext is `"chargers win superbowl"`, encrypted array will be:

```
{10, 14, 8, 6, 13, 0, 6, 4, 26, 23, 3, 18, 26, 4, 22, 5,
                                          0, 6, 9, 19, 23, 1}
```

Decryption

- decryption checks the encryption process. Takes the alphabet string and the array of integers called encrypted, produces the plaintext message e.g. to decode the example above
  - alphabet is:

    ```
    e l v i s p r y a b c d f g h j k m n o q t u w x z ' '
    ```

  - encrypted is:

    ```
    {10, 14, 8, 6, 13, 0, 6, 4, 26, 23, 3, 18, 26, 4, 22, 5,
                                              0, 6, 9, 19, 23, 1}
    ```

  - for each value in the encrypted array, index in to alphabet to produce a character
  - which gives the plaintext:

    ```
    c h a r g e r s ' ' w i n ' ' s u p e r b o w l
    ```

Class specifications

First code and test a class called `Encrypt` that will encrypt messages. This class takes a keyword and a plaintext message and produces the alphabet string and the array of `int` that is the encrypted message. `Encrypt` will have the following instance variables:

```
- String keyword        // e.g. "elvispresley"
- String alphabet       // built from keyword
- String plaintext      // e.g. "chargers win superbowl"
- int encrypted[]       // to be set
```

`Encrypt` will have the following methods:

+ constructor	has parameters to set `keyword` and `plaintext`. First creates `alphabet` from `keyword`. Then produces the array of `int` that is the encrypted message. Here's the pseudocode: set `keyword` call `putKeyword` method to put `keyword` into beginning of `alphabet`

```
call buildAlphabet method to built rest of alphabet

set plaintext
allocate memory for encrypted[] to hold plaintext.length() ints
call encrypt method to do encryption
```

`- putKeyword` puts `keyword` into beginning of `alphabet`, omitting duplicate chars. (Use a local `StringBuffer` in which to build the alphabet, because its size changes). In pseudocode:

```
declare local StringBuffer object alpha and initialize to empty

append first char from keyword to end of alpha
loop an index i for all remaining chars in keyword
        set boolean found to false
        loop an index j in keyword from 0 to less than index i
                if char at i is same as char at j
                        set found to true
        if not found append char at i to alpha
set alphabet from StringBuffer alpha
```

`- buildAlphabet` appends remaining lowercase characters to `alphabet` in order, omitting those that came from `keyword.` In pseudocode:

```
create local StringBuffer object alpha from alphabet
set len to length of alpha

loop char ch from 'a' to 'z'
        set boolean found to false
        loop an index j in alpha from 0 to less than len
                if char at j is same as char ch
                        set found to true
        if not found append char ch to alpha
append space to alpha
set alphabet from StringBuffer alpha
```

The following shows how to loop `char ch` from `'a'` to `'z'`

```
for (char ch = 'a'; ch <= 'z'; ++ch)
        System.out.println(ch); // prints 'a'...'z'
```

`- encrypt` traverses `plaintext`, uses `getIndex` method to convert each `char` to its index in `alphabet`. Pseudocode:

```
loop an index i for all chars in plaintext
```

```
        set encrypted[i] to getIndex(char at i in plaintext)
```

`- getIndex`	has a `char` parameter, searches `alphabet` for the `char` and returns its index
`+ getAlphabet`	returns the `alphabet` string
`+ getEncrypted`	returns `encrypted`
`+ printEncrypted`	traverses `encrypted` and prints it out in Java array format

Now code and test a `Decrypt` class that checks the encryption process. This class takes the alphabet string and an array of `int` that is the encrypted message, produces the plaintext message. `Decrypt` instance variables:

```
- String alphabet       // from Driver
- String plaintext      // to be set
- int encrypted[]       // from Driver
```

`Decrypt` will have the following methods:

+ constructor — has parameters to set `alphabet` and `encypted`. Sets `plaintext`. Pseudocode:

```
set alphabet
set encrypted
call decrypt method to do decryption
```

`- decrypt` — traverses `encrypted` array, indexes into `alphabet` to convert each `int` to a `plaintext` `char`. Pseudocode:

```
loop an index i for all ints in encrypted
        append to plaintext the alphabet char at the index
        stored in encrypted[i]
```

`+ getPlaintext` — returns the `plaintext` string

Required
A `Driver` class is provided that uses your classes to encrypt then decrypt a message. `Driver` may not be altered in any way. `Driver` is available for download at Blackboard, Course Documents, Week 11, Examples, Lab 5.

```
/**
 * Driver for Lab 5.
 *
 * @author Anthony W. Smith
 * @version 6/15/2009
 */
public class Driver
{
    public static void main(String arg[])
    {
        String k = "elvispresley";
        String p = "chargers win superbowl";

        Encrypt encrypt = new Encrypt(k, p);
        System.out.println("Alphabet string:");
        String a = encrypt.getAlphabet();
        System.out.println(a);
        System.out.println("Encrypted message:");
        encrypt.printEncrypted();
        int e[] = encrypt.getEncrypted();

        Decrypt decrypt = new Decrypt(a, e);
        System.out.println("\nDecrypted message:");
        System.out.println(decrypt.getPlaintext());
    }
}
```

Also required, otherwise you cannot score full credit:

- all of your methods must have good Javadoc comments

- automatically and routinely use all the other components of simplicity and clarity, as listed in Blackboard, Course Information, “How labs are graded”

Hints
Before writing your program, work carefully through the `"elvispresley"` encrypt/decrypt example given above, be sure you understand.

Now follow the usual add/compile/test as you go development process:

- design your new algorithms on a piece of paper

- when you're ready to begin coding, download, save and unzip the 'Lab 5' project from Blackboard, Course Documents, Week 11, Examples

- for `String` class, use `charAt()` and `length()` API methods

- for `StringBuffer` class, use appropriate constructors, `append(), toString()` and `length()` API methods.

- for `int` array processing, use `length`, and `[]` to access elements e.g. `encrypted[i]`

- use BlueJ to write and test your methods one at a time as you go…

- …finish one method before starting a next, and so on

- when you have tested your program, in Terminal Window use **Options | Save to file…** to save your output file as `output.txt`

Lab 5 submission

- deadline for this lab is 3 weeks, by end of lab Thursday 11/24. (This is Thanksgiving Day, so you will hand-in your work during class on Tuesday 11/29.)

- as usual, print out the `.java` source files and the output of your program and hand-in to me

- this is a graded lab, so a reminder that you may not copy code from other people

- start work early! Late labs will be penalized