

# CS 225 - Software Design and Development

## Program 4

**Due: Tuesday, September 29, 11:30 PM**

### Objectives

- To gain experience using the Strategy pattern

### Cryptography

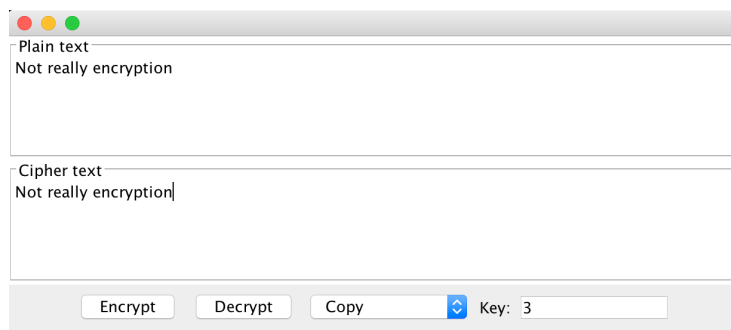
Cryptography provides an excellent example of where multiple strategies can be used to accomplish the same task. Encryption algorithms have existed since ancient times, and there have been many algorithms developed. There are two tasks to perform:

- Encrypt some text to produce a cipher text
- Decrypt some cipher text to produce plain text.

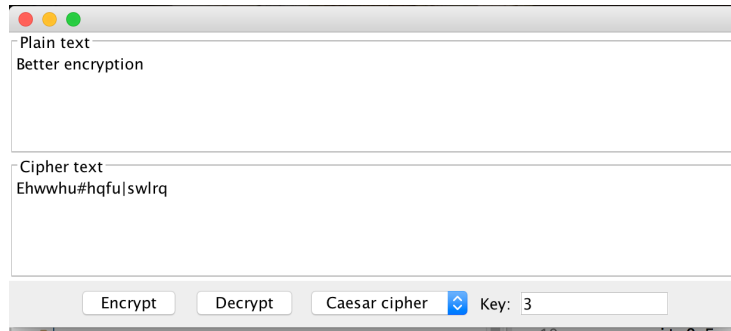
In this assignment, you will use the Strategy pattern to provide a simple application that can encrypt and decrypt text, allowing the user to choose which encryption strategy to use from a menu.

The starter code provides the graphical user interface. You need to define some methods to use the Strategy pattern and to actually do the encryption and decryption.

Here are some snapshots of the cryptography application. The top text area displays unencrypted text. The bottom area displays encrypted text. The user will either type some plain text into the top area, an integer key in the key field, and click the Encrypt button to let the program do the encryption and display the result in the cipher text area or the user will enter some cipher text in the bottom area and click the decrypt button and let the algorithm produce the plain text. The snapshots below show the screen after the encrypt/decrypt button has been clicked so that we can see both encrypted and plain text.



This example shows Copy being the encryption choice. In this case, there really is no encryption being done. Instead, the plain text is just copied to cipher text and vice versa. This would, of course, not be very useful, but it does make for a simple strategy to implement!

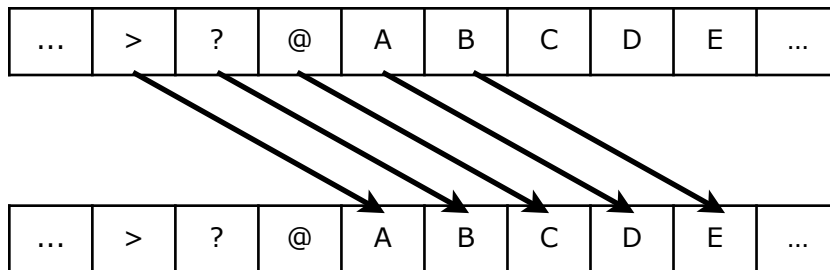


This example uses the Caesar cipher. A Caesar cipher shifts each letter a fixed amount, in this case by 3 letters, since the key is 3. Thus B becomes the letter 3 positions later, or E. e becomes h, etc. This is also not hard to break, but it at least looks encrypted!

Caesar cipher is an example of a substitution cipher, where one letter is substituted for another.

## Caesar cipher

The Caesar cipher is a very simple algorithm. Each letter is shifted a constant amount as shown in the figure below. Here the top row shows an unencrypted character. The arrow connects the character to the character 3 positions later in ASCII, which is the encrypted letter.



ASCII defines the order of characters, with each character being represented by a value in the range of 0 to 255 (an 8-bit value). You can find the complete ASCII table online at many sites, including <http://www.asciitable.com/>.

Java allows you to do arithmetic on characters. In particular, if you have a char variable and add 3 to it, you will get the char value that is 3 positions later in ASCII. Using arithmetic like this, you can implement the Caesar cipher. You will need to be careful to wrap around, so the character with ASCII value 255 gets mapped to the ASCII value 2, for example.

## Scytale Cipher

The Scytale cipher is a cipher that uses transposition rather than substitution, meaning that it changes the order of letters. This algorithm works by first writing the letters in rows with a fixed number of columns, like this:

```
A thir
d algo
rithm
```

Each row contains 6 columns except for the last row, which we will fill in with space characters.

Having done this, we next read the encrypted text off by printing out the columns, like this:

Adr itathlhighmro

## Assignment

For this assignment, you should complete the encryption application. To do this, you will need to create some new classes / interfaces to use the Strategy pattern. While you will need to make some changes in the CryptographyGUI class, you do not need to understand how the GUI is implemented, but it is there for your perusal if you want to check that out!

In the CryptographyGUI class, there are 3 methods that you need to complete. The code currently contains stubs so that it will compile. These methods are:

- selectEncryption - which should set the strategy to use
- encryptWithKey - which should use the selected encryption algorithm
- decryptWithKey - which should use the selected decryption algorithm

Each of these methods currently has a comment that begins:

```
// TODO:
```

to help you find the code you need to change in CryptographyGUI.

You should provide the 3 encryption algorithms shown above (Copy, Caesar, Scytale). Caesar should use the key to determine how many characters to shift by. Scytale should use the key to determine the number of columns to use.

You should also create some JUnit tests to check that your encryption and decryption code works correctly.

Remember to run SpotBugs and checkstyle, too!

## Grading

- (4 points) Good use of the Strategy pattern
- (2 points) Correct implementation of the encryption algorithms
- (1 points) Comments
- (1 point) JUnit testing
- (1 point) Style

## Working with others


You can talk with students in your study group to help you understand the assignment, but you should write your code independently.

## Turning in your work

Turn in your work by committing it to GitHub Classroom.

Another Eclipse gem:

Eclipse conveniently tracks these TODO comments for you. They show up in the left mar-

gin with this icon:  They also show up as clickable blue rectangles in the right margin. If you display the Tasks view they will also show up there as well (Window menu, Show view, Tasks).