Overview: Secret Password

Students may already have been asked to implement the Caesar cipher, or a simple rot-*n* cipher. This activity is very similar and equally cryptographically insecure. In this activity, students are asked to implement a cipher that uses a String as a password instead of the integer "key" *n* of a rot-*n* cipher.

To prepare for this activity, generate enough copies of the following template page for each student to have two wheels (one inner, one outer) and provide scissors and brass tacks for the construction of the wheels.

The students are first asked to play with the Caesar cipher and find out how easy it is to break. Then they are provided with a program that enciphers and deciphers data using a Caesar cipher, and asked to extend it to use a longer key. Asking them to modify an existing program instead of writing their own exposes them to preexisting working code (instead of offering bad code and asking them to fix it). It also shortens the length of the coding part of this activity and puts the focus on the logical thinking behind the coding.

The provided code includes Exceptions thrown on error conditions, which matches with the idea that students might benefit from seeing complete, working preexisting code even if they don't understand all of the syntax invoked. It can be explained that these lines are for error checking, without fully explaining what Exceptions are (a complex topic better left for later in the semester). Alternatively, these could just be taken out from the code.

Prerequisite Material:

This activity requires knowledge of:

- ❑ Loops
- ❑ Math operators
- ❑ String methods






Secret Password

One easy way of keeping messages secret is to use a simple substitution cipher, replacing each occurrence of one letter with another. An easy way to do this is with a Caesar cipher. (This code also happens to be easy to crack.)

Part A. Break the Cipher.

Using the given materials, build your own Caesar cipher wheel. Cut out one whole wheel and one inner wheel, and use a brass tack to connect them.

To use the Caesar cipher, you would pick a word to encipher (your plaintext), and a letter to act as your "key". Set up your wheel for enciphering by aligning the A on the outer wheel with your key on the inner wheel. To encipher your message, find each letter in the word on the outer wheel and replace it with the matching letter on the inner wheel. To decipher, align the key the same way but go from inside out.

Try deciphering the following words with the given keys. Record your answer (the "ciphertext") in the final box.

Ciphertext	**Key**	**Plaintext**
JKLMNOP	J	
KXKKV	X	
WTAADLDGAS	P	

Exchange enciphered words (but not the plaintext or the key) with someone else. Take a few minutes to try to find the other person's word and key.

Part B. Make a Stronger Cipher.

In part A, we used a single letter as the key. Imagine making the cipher stronger by using two letters as the key. Every other time you would switch your wheels between two different positions. This would be harder for a person to keep straight, but a computer would have no problem doing this.

Now imagine extending that to a key of potentially unlimited length. In fact, the only perfectly secure way to encipher your data would be to use a random key that was just as long as your secret message. Modern cryptographic systems try to approximate this, but because they are sometimes predictable, they are still subject to brute-force cracking just like we did in part A.

You are given the code for a program that implements the Caesar cipher from part A. Your task is to modify this program to allow the Key to be more than 1 letter. Look closely at all the classes and decide which method(s) in which class(es) should be changed.