CSCI165 Computer Science II
Lab Assignment
Simple Solitaire Cipher

Objectives:

- Carefully read instructions
- Sequence (array, String) processing with for loops
- Decomposition via static methods
- Unit testing

The Somewhat Simplified Solitaire Encryption Algorithm; aka: The Solitaire Cipher

Overview:

In Neal Stephenson's novel ***Cryptonomicon***, two of the main characters are able to covertly communicate with one another with a deck of playing cards (including the jokers) and knowledge of the Solitaire encryption algorithm, which was created (in real life) by Bruce Schneier. The novel includes the description of the Solitaire algorithm in an appendix.

https://en.wikipedia.org/wiki/Solitaire_(cipher)

For this assignment, we'll ***simplify the algorithm in several ways***. For example, we'll be assuming that we have just two suits (say, hearts and spades) from a deck of cards, plus the two jokers, just to keep things simple. Further, let's assume that the values of the 26 suit cards are 1 to 26 (Ace to King of hearts, followed by Ace to King of spades), including two joker cards labelled "A" and "B". The "A" joker will be numbered 27, and the "B" joker numbered 28. Thus, 15 represents the ***2 of Spades***.

Now that you've got the idea, note that because we are doing this in a computer, we can just use the numbers 1–28 and forget about the suits and ranks.

The Keystream:

The challenging aspect of the Solitaire Cipher is the generation of the ***keystream values***. These values will be used as keys to encrypt or decrypt our messages. Here are the steps used in our variant of the algorithm. Assume that we start with an array of the values from 1–28 as described above. From here on this will be referred to as the ***deck***. Also assume that the cards (numbers in our case) are arranged in some sort of shuffled order.

The order of the deck is important because you need the same order to encrypt and decrypt.

1. **Find the "A" joker** (27). Exchange it with the card beneath (after or under) it in the deck, to move the card down the deck by one position. What if the joker is the last card in the deck? Imagine that the deck of cards is continuous; the card following the bottom card is the top card of the deck, and you'd just exchange them. ***The array needs to act as if it is a loop***. A card taken from the bottom of the deck (end of the array) loops around to be placed on the top of the deck (front of the array). This is slightly different from the true algorithm as descrbed in the Wiki article
2. **Find the B joker** (28). Move it two cards down by performing two exchanges, honoring the **loop** if necessary.

3. **Swap the cards above the first** joker (the one closest to the top of the deck ie. Toward the front of the array) with the cards below the second joker (after the joker in the array). This is called a ***triple cut***.
4. **Take the bottom card from the deck**. Count down from the top card by a quantity of cards equal to the value of that bottom card. (If the bottom card is a joker, let its value be 27, regardless of which joker it is.) Take that group of cards and move them to the bottom of the deck. Return the bottom card to the bottom of the deck.
5. **Look at the top card's value** (which is again 1-27, as it was in the previous step). Put the card back on top of the deck. Count down the deck by that many cards. Record the value of the NEXT card in the deck, but don't remove it from the deck. If that next card happens to be a joker, don't record anything. Leave the deck the way it is, and start again from the first step, repeating until that next card is not a joker.

The value that you recorded in the last step (5) is ***one value of the keystream***, and will be in the range 1 – 26, inclusive (to match with the number of letters in the alphabet). To generate another value, we take the deck as it is after the last step and repeat the algorithm steps 1 - 5.

We need to generate as many keystream values as there are letters in the message being encrypted or decrypted.

An example will really help make sense of the algorithm. Let's say that this is the original ordering of our half–deck of cards. (Remember . . . jokers are 27 (A) and 28 (B)):

```
1 4 7 10 13 16 19 22 25 28 3 6 9 12 15 18 21 24 27 2 5 8 11 14 17 20 23 26
```

Step 1: Swap the A joker (27) with the value following it. So, we swap 27 and 2:

```
1 4 7 10 13 16 19 22 25 28 3 6 9 12 15 18 21 24 2 27 5 8 11 14 17 20 23 26
                                                ^^^^
```

Step 2: Move the B joker (28) two places down the list. It ends up between 6 and 9:

```
1 4 7 10 13 16 19 22 25 3 6 28 9 12 15 18 21 24 2 27 5 8 11 14 17 20 23 26
                        ^^^^^^
```

Step 3: Do the triple cut. Everything above (before, in the deck) the first joker (28 in this case) goes to the bottom of the deck, and everything below (after, in the deck) the second joker (27) goes to the top. This will result in the following.

```
5 8 11 14 17 20 23 26 28 9 12 15 18 21 24 2 27 1 4 7 10 13 16 19 22 25 3 6
^^^^^^^^^^^^^^^^^^^^^                          ^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

Step 4: The bottom card is 6. The first 6 cards of the deck are 5, 8, 11, 14, 17, and 20. They go just ahead of 6 at the bottom end of the deck:

```
5 8 11 14 17 20 23 26 28 9 12 15 18 21 24 2 27 1 4 7 10 13 16 19 22 25 3 6

23 26 28 9 12 15 18 21 24 2 27 1 4 7 10 13 16 19 22 25 3 5 8 11 14 17 20 6
^^^^^^^^^^^^^^^^
```

Step 5: The top card is 23. Thus, our generated keystream value is the 24th card, which is 11.

```
23 26 28 9 12 15 18 21 24 2 27 1 4 7 10 13 16 19 22 25 3 5 8 11 14 17 20 6
```

You have just generated a single keystream value. You need ***original_message.length*** keystream values.

OK, so what do you do with all of those keystream values? The answer depends on whether you are encoding a message or decoding one.

Encoding:

To encode a message with the Solitaire Cipher, remove all non–letters from the plain text and convert any lowercase letters to uppercase. (If you wanted to be in line with traditional cryptographic practice, you'd also divide the letters into groups of five.) Convert the letters to numbers (A=1, B=2, etc.). Use the Solitaire Cipher to generate the same number of keystream values as are in the message. Add the corresponding pairs of numbers, modulo 26. Convert the numbers back to letters, and you're done.

Decoding:

Decryption is just the reverse of encryption. Start by converting the message to be decoded to numbers. Using the ***same deck ordering*** as was used to encrypt the message originally, generate enough keystream values. (Because the same starting deck of cards was used, the same keystream will be generated.) Subtract the key stream values from the message numbers, again modulo 26. Finally, convert the numbers to letters and read the message.

Let's give it a try. The message to be sent is this

```
Cryptography, I love!
```

Removing the non-letters and capitalizing gives us

```
CRYPTOGRAPHYILOVE
```

This message has 17 letters which is not a multiple of 5, so we will need to pad out to 20 characters with Xs.

```
CRYPTOGRAPHYILOVEXXX
```

Next, convert the letters to numbers based on 1 - 26

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**
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**

C	R	Y	P	T	O	G	R	A	P	H	Y	I	L	O	V	E	X	X	X
3	18	25	16	20	15	7	18	1	16	8	25	9	12	15	22	5	24	24	24

Rather than actually generating a sequence of 20 keystream values for this example, let's just pretend that we did. Our keystream is below. Each one of the keystream values will be mathematically applied to the numbers generated from the letters of the original message.

```
21  6   2   19  15  18  12  23  23  5   1  7   14  6   13  1   26  16  12  20
```

Add the two groups of numbers together pairwise and modulo 26

	21	6	2	19	15	18	12	23	23	5	1	7	14	6	13	1	26	16	12	20
+	3	18	25	16	20	15	7	18	1	16	8	25	9	12	15	22	5	24	24	24
	24	24	27	35	35	33	19	41	24	21	9	32	23	18	28	23	31	40	36	44
%26	24	24	1	9	9	7	19	15	24	21	9	6	23	18	2	23	5	14	10	18

And then convert back to letters

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**
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**

24	24	1	9	9	7	19	15	24	21	9	6	23	18	2	23	5	14	10	18
X	X	A	I	I	G	S	O	X	U	I	F	W	R	B	W	E	N	J	R

Encoded Message: XXAIIGSOXUIFWRBWENJR

Decryption:

Convert the encrypted message's letters to numbers based on the plain text alphabet positions

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**
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**

X	X	A	I	I	G	S	O	X	U	I	F	W	R	B	W	E	N	J	R
24	24	1	9	9	7	19	15	24	21	9	6	23	18	2	23	5	14	10	18

Generate the same keystream using the ***same initial deck organization*** as the encryption routine

```
21  6   2   19  15  18  12  23  23  5   1  7   14  6   13  1   26  16  12  20
```

Subtract the keystream values from the message numbers. To deal with the reversal of modulo 26 from the encoding, ***just add 26 to the top number if it is equal to or smaller than the bottom number***.

	24	24	1	9	9	7	19	15	24	21	9	6	23	18	2	23	3	14	10	18
-	21	6	2	19	15	18	12	23	23	5	1	7	14	6	13	1	26	16	12	20
	3	18	25	16	20	15	7	18	1	16	8	25	9	12	15	22	3	24	24	24

Now convert the numbers to letters and you are back to your original message!

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
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

3	18	25	16	20	15	7	18	1	16	8	25	9	12	15	22	7	24	24	24
C	R	Y	P	T	O	G	R	A	P	H	Y	I	L	O	V	E	X	X	X

Decoded Message: CRYPTOGRPHYILOVEXXX

Tasks: FOCUS ON TESTING

1. You have been provided with the file: ***SolitaireCipher.java***. This file contains
    a. A couple of Solitaire Cipher methods that have been implemented and tested for you. These implementations are based on the instructions in this document.
    b. A variety of method signatures that you must implement. These methods are defined in a way to illustrate the difference between passing arrays and String into methods. Notice that the methods that manipulate arrays are void, due to the reference passing. The methods that manipulate Strings are non-void due to the immutability of String objects.
2. You have been provided with the file: ***SolitaireTester.java.*** This file contains tester methods for the implemented method in ***SolitaireCiper.java***
3. Implement the method stubs based on the descriptions in the instructions. Focus on breaking problems down into single tasks, defining methods for the task and then writing tester methods to ensure that the method functions correctly. Feel free to write as many methods as you need but I will be expecting the signatures that are provided to you as I have tester functions for those.
4. Include simple ***Javadoc*** style method documentation as shown in the examples. We will be developing this style more as we proceed.
5. For each method you define and implement include tester methods as shown in the examples. Include 3 test cases for each. Advisable to include more.
6. Write as many helper methods as you need. There is a section marked **HELPERS** in the code. Place them there.
7. You do not need to worry about reading and writing to files. You can simply encode and decode short messages passed through ***command line arguments***

Expected Program Functionality:

- Run the program by passing in a message to be enciphered (encrypted, encoded . . . etc) as a command line argument.
- Include a flag **e** (encipher) and **d** (decipher)
- Print the encoded message to the console. The reverse should also work.
- What about the deck? You have to make some decisions here.

Submission:

- SolitaireCipher.java
- SolitaireTester.java

EXTRA CHALLENGES: These are not required but the adventurous among you may enjoy the challeng.

1. Modify the ***joker exchange*** methods to fit with the actual algorithm as described in the Wiki article. This particularly pertains to the looping of a card at the end of deck to the front of the deck. You'll notice the provided implementation does a simple exchange, moving the top card to the end of the deck and the joker to the front instead of inserting the Joker into its proper location at the front of the deck. (The Joker cannot be the first card according to the true algorithm)
2. Encrypt and decrypt files, adhering to the cryptographic standard of grouping characters in chunks of 5 when writing to disk.
3. Design a scheme to serialize a generated deck order so it can be used in the decrypt routine. For example
    a. A file is encrypted using a randomly ordered deck.
    b. The file would then need to be decrypted using the same deck ordering
    c. Design a way for the deck to be included in the file, but not in plain text. Perhaps the decryption routine begins with deciphering the deck and then using that deck to generate the keystream and perform decryption.