Group Assignment 1

Background

Topics:
- OOP
- Methods
- Strings
- Arrays
- ArrayLists

Description:
For this assignment, you will code a simple encryption application. The application encrypts words by mapping a list of the twenty-six alphabetic characters to a list of twenty-six corresponding symbols using the lists' indices.

0	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
!	@	#	$	^	&	*	(	)	_	-	+	=	?	,	{	}	[	]	/	|	;	:	.	<	>

symbols list

0	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
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

alphabets list

For example, the word Elvis, would encrypt as follows:

E -> ^
l -> +
v -> ;
i ->)
s ->]

So, Elvis would encrypt to: ^+;)]

Group Assignment 1

Instructions

You will start with the following Encryption class definition:

```
import java.util.ArrayList;

class Encryption {
  ArrayList<Character> symbols = new  ArrayList<Character>();
  ArrayList<Character> alphabets = new ArrayList<Character>();

  public Encryption() {
    symbols.add('!');
    symbols.add('@');
    symbols.add('#');
    symbols.add('$');
    symbols.add('^');
    symbols.add('&');
    symbols.add('*');
    symbols.add('(');
    symbols.add(')');
    symbols.add('_');
    symbols.add('-');
    symbols.add('+');
    symbols.add('=');
    symbols.add('?');
    symbols.add(',');
    symbols.add('{');
    symbols.add('}');
    symbols.add('[');
    symbols.add(']');
    symbols.add('/');
    symbols.add('|');
    symbols.add(';');
    symbols.add(':');
    symbols.add('.');
    symbols.add('<');
    symbols.add('>');

    for(char letter = 'a'; letter <= 'z'; letter++) {
      alphabets.add(letter);
    }
  }
}
```

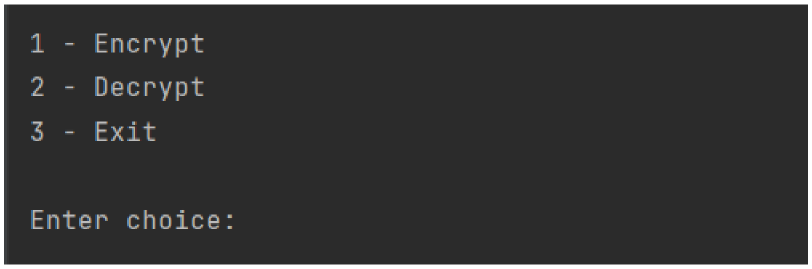
Group Assignment 1

This creates two Java ArrayLists. The first list contains twenty-six symbols (i.e. ~,@,#,$,%, etc.), one symbol in each cell. The second list contains the twenty-six lower case letters of the alphabet (a – z), one letter in each cell. From there you'll create the following six (6) Encryption class methods:

1) Add a method which takes in an int (integer) and returns the alphabet stored at that position, i.e. 5 would return f.
2) Add a method which takes in an alphabetic character (char) and returns the index (int) of the character in the alphabets list, i.e. a would return 0, b would return 1.
3) Add a method which takes in an int (integer) and returns the symbol stored at that position, i.e. 5 would return &
4) Add a method which takes in a symbol (char) and returns the index (int) of the symbol in the symbols list, i.e. ! would return 0, @ would return 1.
5) Add a method which takes in a plain-text string and returns the encrypted version of that string, i.e. Dwags would return $:!*]
   - The method should convert the plain-text string to lowercase, hint: see Java method ".toLowerCase()"
   - The method should process each character in the plain-text string individually, encrypting each character and building a new string of encrypted characters (hint: see Java method "toCharArray()", and the newly created methods above)
   - If an invalid alphabet character is found, the following string should be returned: "Error: Invalid Character"
6) Add a method which takes in an encrypted string and returns the decrypted version of that string, i.e. $:!*] would return dwags
   - The method should process each symbol in the encrypted string individually, decrypting each symbol and building a new string of decrypted characters (hint: see Java method "toCharArray()", and the newly created methods above)
   - If an invalid symbol is found, the following string should be returned: "Error: Invalid Symbol"

The driver class will contain the following:

1. Create an object of the class Encryption.
2. Prompt the user with the following menu:



```
1 - Encrypt
2 - Decrypt
3 - Exit

Enter choice:
```

3. If the user enters 1, prompt the user to enter and read in a plain-text message, encrypt the entered message, and print the encrypted message.
4. If the user enters 2, prompt the user to enter and read in an encrypted message, decrypt the entered message, and print the decrypted message.
5. If the user enters 3, terminate the program.
6. If the user enters any alpha-numeric or non-alpha-numeric characters other than 1, 2, or 3, the following error message should display: Error: Please enter valid input, and the user should be allowed to reenter another choice, i.e. **entering an invalid choice should not terminate the program.**
7. **See Example Screenshots below for additional information.**

Submission Instructions

You must rename your .java files to .txt and submit the following items to the D2L dropbox before the deadline:

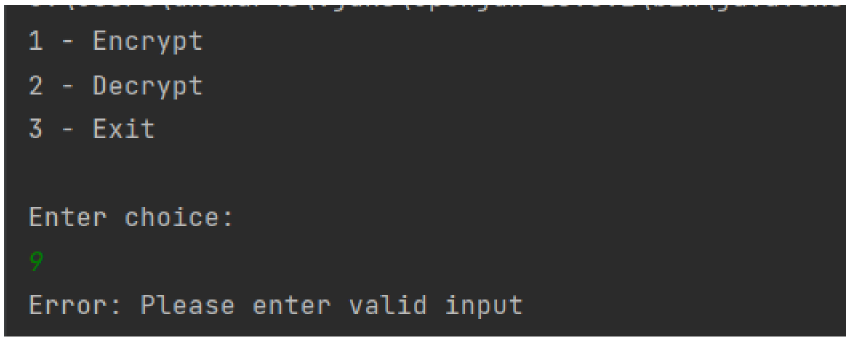
1. Encryption.txt
2. EncryptionTest.txt

Grading Criteria:

1. successfully copied and pasted the Encryption class (2 pts)
2. return_alphabet_index method (6 pts)
3. return_alphabet *method* (6 pts)
4. return_symbol_index method (6 pts)
5. return_symbol *method* (6 pts)
6. encrypt method (25 pts)
7. decrypt method (23 pts)
8. driver program (20 pts)
9. entering invalid choice doesn’t terminate program (6 pts)

Group Assignment 1

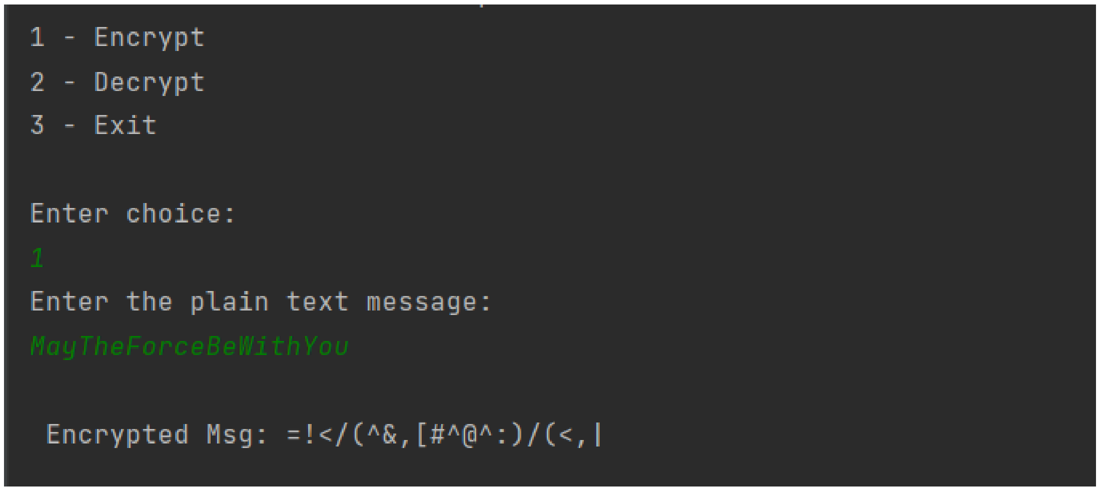
Example Screenshots



```
1 - Encrypt
2 - Decrypt
3 - Exit

Enter choice:
9
Error: Please enter valid input
```

Example 1: User entered invalid choice. The only valid choices are 1, 2, or 3.

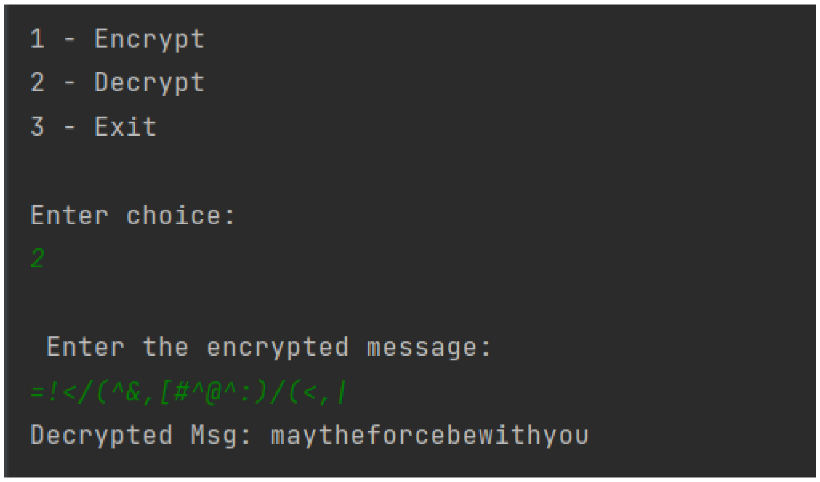


```
1 - Encrypt
2 - Decrypt
3 - Exit

Enter choice:
1
Enter the plain text message:
MayTheForceBeWithYou

 Encrypted Msg: =!</(^&,[#^@^:)/(<,|
```

Example 2: User entered valid choice and entered valid plan-text message to be encrypted.

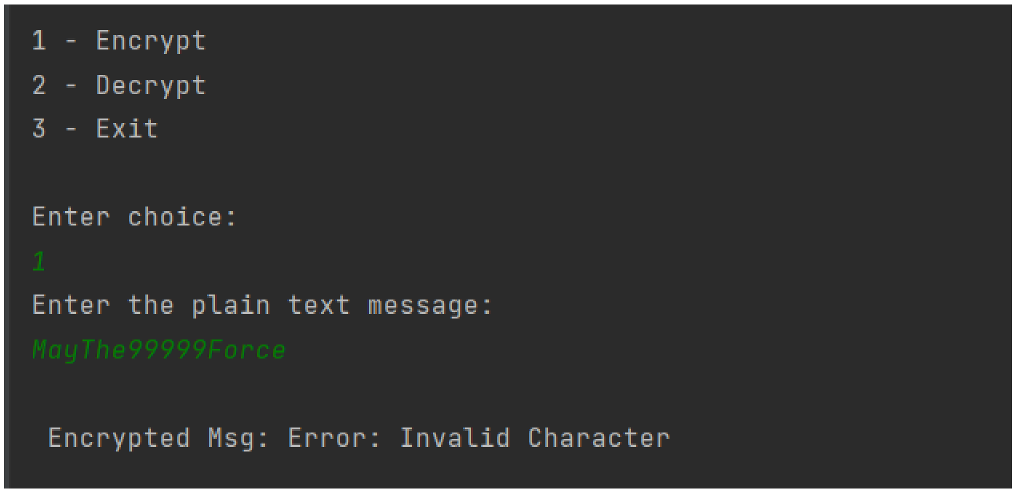


```
1 - Encrypt
2 - Decrypt
3 - Exit

Enter choice:
2

 Enter the encrypted message:
=!</(^&,[#^@^:)/(<,|
Decrypted Msg: maytheforcebewithyou
```

Example 3: User entered valid choice and entered valid encrypted message to be decrypted.

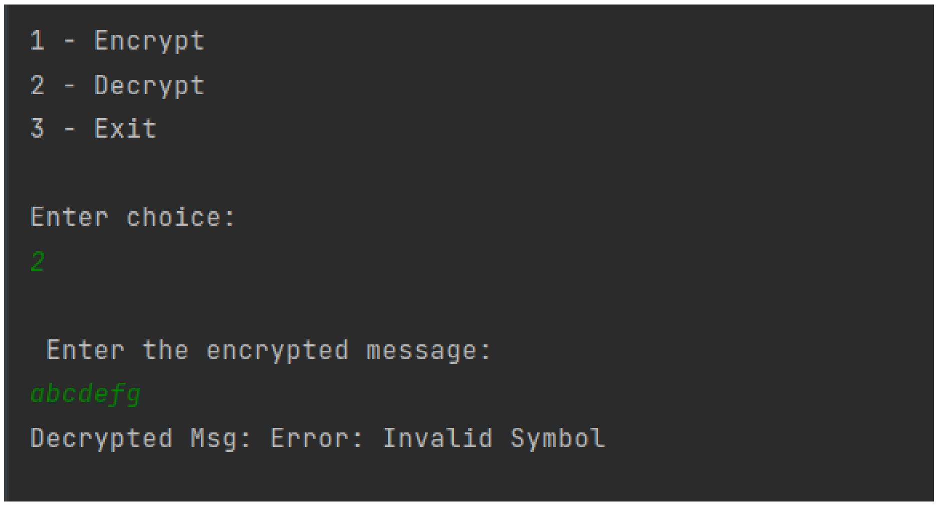


```
1 - Encrypt
2 - Decrypt
3 - Exit

Enter choice:
1
Enter the plain text message:
MayThe99999Force

 Encrypted Msg: Error: Invalid Character
```

Example 4: User entered valid choice but entered invalid plain-text message. Plain-text messages should only contain alphabetic characters, as shown in “alphabets” list.



```
1 - Encrypt
2 - Decrypt
3 - Exit

Enter choice:
2

 Enter the encrypted message:
abcdefg
Decrypted Msg: Error: Invalid Symbol
```

Example 5: User entered valid choice but entered invalid encrypted message. Encrypted messages should only contain non-alphabetic characters, as shown in “symbols” list.