Index	Reference Table	Shift Table #1 (Offset Character **B**)	Shift Table #2 (Offset Character **F**)
0	A	/	+
1	B	A	,
2	C	B	-
3	D	C	.
4	E	D	/
5	F	E	A
6	G	F	B
7	H	G	C
8	I	H	D
9	J	I	E
10	K	J	F
11	L	K	G
12	M	L	H
13	N	M	I
14	O	N	J
15	P	O	K
16	Q	P	L
17	R	Q	M
18	S	R	N
19	T	S	O
20	U	T	P
21	V	U	Q
22	W	V	R
23	X	W	S
24	Y	X	T
25	Z	Y	U
26	0	Z	V
27	1	0	W
28	2	1	X
29	3	2	Y
30	4	3	Z
31	5	4	0
32	6	5	1
33	7	6	2
34	8	7	3
35	9	8	4
36	(	9	5
37	)	(	6
38	*	)	7
39	+	*	8
40	,	+	9
41	-	,	(
42	.	-	)
43	/	.	*

Requirement:

You are to write an encoder that takes in a plaintext and encode it to another obfuscated string. The logic of the encoding / decoding is given below:

Logic:

Choose **any** character in the reference table as the offset. The **first character** of the encoded message will be the offset character. Any character not in the reference table will mapped back to the same character.

For example, if the offset character is **B**, the entire table will shift by **1** element down (Refer to Shift Table #1). Thus, given the plaintext *HELLO WORLD,* it will be encoded as ***B****GDKKN VNQKC:*

	H	E	L	L	O		W	O	R	L	D
B	G	D	K	K	N		V	N	Q	K	C

Let's take **F** as the offset character for another example. The entire table will shift **5** elements down (Refer to Shift Table #2). Given the same plaintext, the encoded message will be:

	H	E	L	L	O		W	O	R	L	D
F	C	/	G	G	J		R	J	M	G	.

To decode it, you need to take the first character for offset and match it backwards to get the original plaintext.

Constraints

The solution must implement the following 2 methods:
public String encode (String plainText);
public String decode (String encodedText);

Bonus

The solution should also demonstrate the concept of **OOP**.

Note: **B** and **F** are used for illustration only. Your code should be able to use any of the 44 characters in the reference table as offset.