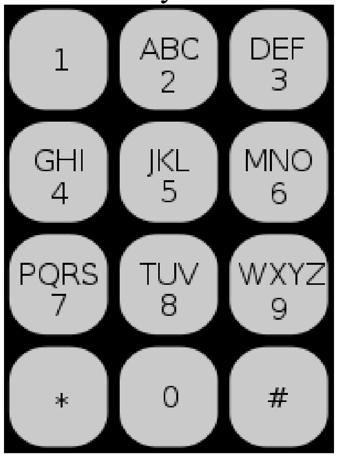
Homework #3

Due: **10/1/18 at 10:00 PM**
Late Collections: **No Late Collections**
Program Files (2): **TextDecoder.java, TextMessage.java**

In the 1990's and early 2000's, few cell phones had a full keyboard and touchscreen were not available. As such, users had to to write text messages using the numeric keypad. Each key is assigned multiple letters, and one way to specify a letter is to press the same key in rapid succession, cycling through all letters on that key. Consider the phone keypad given below:



To get the letter 'A', you press 2 once, if you press 2 twice, you get 'B', and if you press it three times you get 'C'. If you want to produce two letters in sequence that use the same key, then you must wait a short time after the first letter is chosen before pressing the key again. Pressing 0 produces a space and pressing 1 produces a period. Note that although most keys have three letters, *both 7 and 9 have four letters*. The table below summarizes the inputs:

Letter	**Keys**
A	2
B	2,2
C	2,2,2
D	3
E	3,3
F	3,3,3
G	4
H	4,4
I	4,4,4
J	5
K	5,5
L	5,5,5
M	6
N	6,6
O	6,6,6

Letter	**Keys**
P	7
Q	7,7
R	7,7,7
S	7,7,7,7
T	8
U	8,8
V	8,8,8
W	9
X	9,9
Y	9,9,9
Z	9,9,9,9
.	1
space	0
pause	' '

Note, we are using the space character to indicate a pause between entering keys. Your task is to write a program that can read a series of text messages entered as a series of keys from a file, and then decode each message. The file name will be specified as a command-line argument. Each line of the file will contain the number of the recipient followed by the key presses. In general the key presses will be sequences of digits (*no commas)*, but there may be the occasional space representing a pause.

This homework tests your ability to use methods of the String class, to use command-line arguments, and to use the Scanner class with Files. You should also use good object-oriented design within your methods. You need to write two classes, which are described using UML below:

TextMessage	A single text message
-recipient:String	Phone number of the recipient
-keyPresses:String	The series of key presses for the message.
+TextMessage(recipient:String, keyPresses:String)	Initialize fields using parameter values.
+getRecipient():String	Get method for **recipient**
+getDecodedMessage():String	Return the string that represents the decoded version of the message.

TextDecoder	A class that can read a series of text messages from a file and then print out the decoded versions of them.
-messages:TextMessage[]	An array of TextMessages. If there are fewer messages than the size of the array, nulls appear at the end of the array.
-msgCount:int	The number of messages stored in the decoder.
+TextDecoder()	Initializes the **messages** array to be able to hold up to 10 messages and sets the **msgCount** to 0.
+readMessagesFromFile(msgFile:File):void	Reads messages from the file specified by the **msgFile** object, updating **messages** and **msgCount** appropriately.
+printMessages():void	Prints all of the **messages** using the form: "*recipient*:[tab]*decoded-message*". See the example output on the next page.
+main(args:String[]):void	Checks if there is one command-line argument and exits with a helpful error message if not. Otherwise, reads a set of messages from the specified file and then prints out the decoded messages in the form above.

Consider the following sample input file:

```
6107586533 44335557075557777
4848675309 53366 66999044404666804448
6107584096 94466602777330999666887770223377778077778 883 336687777
```

Assuming the file is named **messages.txt** and the program is started by the following command:

```
> java TextDecoder messages.txt
```

A sample run of the program should look like this:

```
6107586533:   HELP PLS
4848675309:   JENNY I GOT IT
6107584096:   WHO ARE YOUR BEST STUDENTS
```

You may assume that the input file will have the correct format and that no input file will have more than 10 text messages. Each message will consist only of the digits 0 through 9 and the space (to indicate pauses). The recipient's number will not have any whitespace or punctuation.

At a minimum provide a Javadoc comment explaining what each method does, and another one that summarizes the class. Include additional comments for lines that are especially complicated. At the top of the program include a comment with the following form:

```
/*
CSE 17
Your name
Your user id
[if you used a tutor, provide his/her contact information here]
Homework #3      DEADLINE: October 1, 2018
Program: Text Messaging
*/
```

Submission:
If you are using an IDE other than DrJava, make sure that your program is not using packages. Otherwise, you will lose points on your submission.
Once the program compiles, runs, and has been tested to your satisfaction, upload both .java files to Course Site. To do so, click on the name of the assignment in the Course Site page, and then press the "Add submission" button at the bottom of the next page. Drag and drop each file into the area under "File submissions". If necessary, you can update your submission at any time before the deadline passes. Be very careful to ensure that you named the class and files correctly, including using the correct case for all letters in the names. Recall, your main method should be in the TextDecoder class.