CSCI165 Computer Science II
Discussion Problem: Week Three
Code Due: Push to repo before 2/21/2021 11:59 pm

There are two problems in this homework set. They are related but I do want to see them solved separately. Multiple source code files are fine but they can be combined. If you do combine them make it incredibly clear which problem is executing by including console messages.

Discussion: Use this problem as basis for this week's discussion post. Help each other out, share problem solving ideas and code. Give suggestions. We are going to be doing quite a bit of ***sequence processing*** in the next couple of weeks. This small linear example will assist you in preparing for next week's lab and the following two weeks will build upon these concepts.

Problem Description
In a certain code language called **KenSpeak™**, individual, base 10 numerals are represented by either a symbol or a letter. This process is called ***transliteration.***

According to www.vocabulary.com: *Transliteration* is the process of transferring a word from the alphabet of one language to the alphabet of another language

You will transliterate base10 numbers to **KenSpeak™.** A Sample code mapping is shown in the following table

Numeral	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Symbol Code	*****	**B**	**E**	**A**	**@**	**F**	**K**	**%**	**R**	**M**

For example, 487692 is encoded into KenSpeak™ as @R%KME

Problem 1:

- ***Using a command line argument***, write a program that will allow the user to enter a $base_{10}$ number of any length.
- Transliterate the number to **KenSpeak™** using the table above and display the encoded String.
- You will not know how long the entered number will be so you can either
    - Treat it as a String and use an explicit **for loop** based on the **length** of the input.
    - Leave it as a number and separate the digits using division and a while loop. You'll need to find the appropriate power of 10 to begin this operation
- Use this problem as an incremental step in solving and verifying your approach for problem 2

Helpful Hints:

- API research: String, Character, Scanner, FileWriter
- It **may** help to define a static helper method called (not required. We'll be covering methods detail next week)
  - **public static String translate(String original)**

- Define a String called codes and assign the values of the Symbol Code table above
  - **String codes = "*BEA@FK%RM";**
  - You could also define this in a primitive char array
- Now you have a **correlation between the index values of the "codes" String and the numerals** that are being translated. You essentially have that table built and loaded into memory. **You are required to use this approach for the assignment.**
- You can convert a character representation of a digit to its integer equivalent in the following way, using the **Character** class
  - **char c = '1'; // c == ASCII 49**
  - **int a = Character.getNumericValue(c);**
  - **now "a" equals integer 1**
  - Notice that this is not simply performing a type cast to get the ASCII/Unicode value

Requirements:

- I do not want to see explicit if statements of the following form. Code of this format will immediately receive a 50% reduction in grade.

  - ```
    if(input.charAt(x) == '1')
        encodedString += "B";
    ```

- **Rationale**: We want to write code that could easily respond to code sets of different lengths and characters. Embedding the mapping of the character and digit into our decision logic makes for code that cannot be easily updated.

- Use the mapping of ***character cussio*** as described above

Problem 2:

- Write a program that reads the file ***numbers.txt*** one line at a time. It should encode the numbers into **KenSpeak™** and write the encoded values to a new file, called: ***encodedNumbers.txt***
- The first line in the file will be the symbol code for the transliteration. Assume that every file that you run through this process will begin with the transliteration symbols.
    ```

Cool Extension

- Use random numbers to generate a random 10 character transliteration table. Take a look at the ASCII table to find appropriate ordinal ranges and ask Java to give you random numbers in those ranges.
- Transliterate two and from **KenSpeak™**. The “from” portion is fairly simple once you wrap your head around to the “to” portion.