Week 07 - Strings & Enums

Topics covered in this week

- Strings
- Enums

Reading material:

https://www.baeldung.com/java-11-string-api

Help tools:

https://regexr.com/

Homework

Difficulty	Problem	Notes

MEDIUM	Validate the following fields, using regular expressions:
	• email address - validation rules:
	  ◦ The email format is '[email]@[app].[domain]'
	  ◦ The [email] should not contain any other `@`;
	  ◦ The [domain] is either 2 or 3 characters.
	• password - validation rules:
	  ◦ at least 10 characters long
	  ◦ should contain at least 1 upper case
	  ◦ should contain at least 1 lower case
	  ◦ should contain at least 1 special character: !"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
	• phone number - validation rules:
	  ◦ phone number can either contain country code or not
	  ◦ country code is either starting with '+' or '00'. Ex: for Romania, country code can be +40 or 0040	

EASY	Create custom enums for the days of the week and months of the year. Given a date object, output the corresponding enums, along with the year of that date object.	
EASY	To *"titlelize"* a string means to change the first letter of each word in the string to upper case (if it is not already upper case) - ignoring some words. For example, a capitalized version of *"Now is the time to act!"* is *"Now Is the Time to Act!"*. Write an application named that will print a *titlelized* version of a string to standard output. Words to ignore: *the, a, to, in, of*	TDD: Please use the attached project and write the implementation, and make sure all tests pass. Hint: Maven.txt See week 4.p2
MEDIUM	Write an application that uses random number generation to create sentences. Use four arrays of strings called *article*, *noun*, *verb* and *preposition*. Create a sentence by selecting a word at random from each array in the following order: *article*, *noun*, *verb*, *preposition*, *article* and *noun*. As each word is picked, concatenate it to the previous words in the sentence. The words should be separated by *spaces*. When the final sentence is output, it should start with a *capital letter* and end with a period. The program should generate 20 sentences and output them in console. The arrays should be filled as follows: the article array should contain the articles "*the*", "*a*", "*one*", "*some*" and "*any*"; the noun array should contain the nouns "*boy*", "*girl*", "*dog*", "*town*" and "*car*"; the verb array should contain the verbs "*drove*", "*jumped*", "*ran*", "*walked*" and "*skipped*"; the preposition array should contain the prepositions "*to*", "*from*", "*over*", "*under*" and "*on*". After the preceding program is written, modify the program to produce a short story consisting of several of these sentences.	Hint: you can use java.util.Random class

MEDIUM	***(Pig Latin)*** Write an application that encodes English language phrases into pig Latin. *Pig Latin* is a form of coded language often used for amusement. Many variations exist in the methods used to form pig Latin phrases. For simplicity, use the following algorithm: To form a pig Latin phrase from an English language phrase, tokenize the phrase into words with an object of class StringTokenizer. To translate each English word into a pig Latin word, place the first letter of the English word at the end of the word and add the letters "*ay*." Thus the word "*jump*" becomes "*umpjay*," the word "*the*" becomes "*hetay*" and the word "*computer*" becomes "*omputercay*." Blanks between words remain as blanks. Assume the following: The English phrase consists of words separated by blanks, there are no punctuation marks and all words have two or more letters. Method printLatinWord should display each word. Each token returned from nextToken is passed to method printLatinWord to print the pig Latin word. Enable the user to input the sentence. Keep a running display of all the converted sentences in console.	