SDP ASSIGNMENT TASK 2

Name : Rahat Hossain

Roll : 32

**Problem Description:**

*Sentence Generator is a program that generates sentences using English words. All sentence*

*generators have an internal vocabulary (one vocabulary for each sentence generator), which is*

*initially empty and can be updated by adding some words. All sentence generators are able to*

*generate sentences according to some rules. You have to implements following three types of*

*sentence generators:*

*1. Random Sentence Generator - RSG*

*When a word is added to RSG, before storing it in internal vocabulary, the word is*

*converted to lowercase. RSG generates new sentences by randomly picking random*

*amount of words from its internal vocabulary and concatenating them using single space*

*between the words.*

*2. Sorted Sentence Generator - SSG*

*Before adding a word into its internal vocabulary, SSG also converts the word to*

*lowercase. Like RSG, SSG picks up the words randomly. The only difference is that it*

*sorts these words before the concatenation.*

*3. Ordered Sentence Generator - OSG*

*OSG is different from both RSG and SSG. A word, before adding into internal*

*vocabulary, will be converted to upper case and reversed. OSG concatenates all of the*

*words in the same order they have been added to the vocabulary.*

*Finally, after creating these sentence generators you have to create a sentence generator*

*application(console/gui/etc), in which user can choose a particular sentence generator from a*

*menu to create a sentence. Moreover, using the same menu user can also give input to add new*

*words in the vocabulary.*

**Project stack**:

* Python

**Classes used:**

* SentenceGenerator: Abstract class for sentence generator classes
* CaseStrategyInterface: Word Case converting class interface.
* LowerCaseStrategy: Lower case converter class.
* UpperCaseAndReversedStrategy: Upper case converter and reverser class
* SentenceConstructionStrategyInterface: Interface for sentence constructor strategy classes.
* RandomsentenceConstruction: Random sentence constructor class.
* SortedsentenceConstruction: Sorted sentence constructor class.
* OrderedsentenceConstruction: Ordered sentence constructor class.
* RandomSentenceGenerator: Main random sentence generator class.
* SortedSentenceGenerator: Main sorted sentence generator class.
* OrderedSentenceGenerator: Main ordered sentence generator class.

**Assumptions:**

1. SentenceGenerator:
   1. Acts as a parent class for all the sentence generator classes.
   2. Has variable wordList(stores all the words that are given as input). It also has methods: caseStrategy(converts the word into different cases), sentenceConstructionStrategy(constructs a single sentence from the wordList using a defined algorithm), add(appends a word to the wordList).
   3. Setter method for caseStrategy and sentenceConstructionStrategy is provided so that the strategies can be redefined at the runtime without manual coding.
2. CaseStrategyInterface:
   1. Acts as an abstract interface for the case-strategy classes.
   2. Defines a caseStrategy() abstract method that takes a string as input and returns a converted string.
3. LowerCaseStrategy:
   1. Inherited from CaseStrategyInterface.
   2. Redefines caseStrategy() method to make the string lower-case.
4. UpperCaseAndReverseStrategy:
   1. Inherited from CaseStrategyInterface.
   2. Redefines caseStrategy() method to make the string lower-case and reversed.
5. SentenceConstructionStrategyInterface:
   1. Acts as an abstract interface for the sentence-constructor classes.
   2. Defines a sentenceConstructionStrategy() abstract method to construct a sentence from a word-list using an algorithm.
6. RandomsentenceConstruction:
   1. Inherited from SentenceConstructionStrategyInterface.
   2. Redefines sentenceConstructionStrategy() method that takes a list of words as input and returns a sentence which is constructed by words taken randomly from the list.
7. SortedsentenceConstruction:
   1. Inherited from SentenceConstructionStrategyInterface.
   2. Redefines sentenceConstructionStrategy() method that takes a list of words as input and returns a sentence which is constructed by words that are picked randomly from the list and sorted accordingly.
8. OrderedsentenceConstruction:
   1. Inherited from SentenceConstructionStrategyInterface.
   2. Redefines sentenceConstructionStrategy() method that takes a list of words as input and returns a sentence created from those words.
9. RandomSentenceGenerator:
   1. Inherited from SentenceGenerator.
   2. LowerCaseStrategy() and RandomSentenceConstruction() are set as caseStrategy and sentenceConstructorStrategy respectively.
10. SortedSentenceGenerator:
    1. Inherited from SentenceGenerator.
    2. LowerCaseStrategy() and SortedSentenceGenerator() are set as caseStrategy and sentenceConstructorStrategy respectively.
11. OrderedSentenceGenerator:
    1. Inherited from SentenceGenerator.
    2. UpperCaseAndReverseStrategy() and OrderedSentenceGenerator() are set as caseStrategy and sentenceConstruction respectively.
12. Console procedure:
    1. The program gives options to choose from Random Sentence Generator, Sorted Sentence Generator and Ordered Sentence Generator.
    2. User chooses the desired option.
    3. Desired sentence generator instance is declared.
    4. User is asked to input the amount of words he/she is going to use for the sentence generation.
    5. User inputs all the words accordingly. Words are given inside the Sentence Generator Instance.
    6. User is provided the output sentence using the defined strategies for the desired sentence generator. [**NB. The output sentence is never empty.**]

**Overall:**

Strategy design pattern is used for this problem.