Free Response Question String Scramble A

For additional practice with iteration, you will work through a sample AP CS-A Java exam question that asks you to develop an algorithm for scrambling the letters of a string. Be sure to give an answer that uses iteration.

Work through Free Response Question [**String Scramble A**](http://interactivepython.org/runestone/static/JavaReview/LoopBasics/stringScrambleA.html) and in your ScrambledStrings class; write the method scrambleWord. Note:

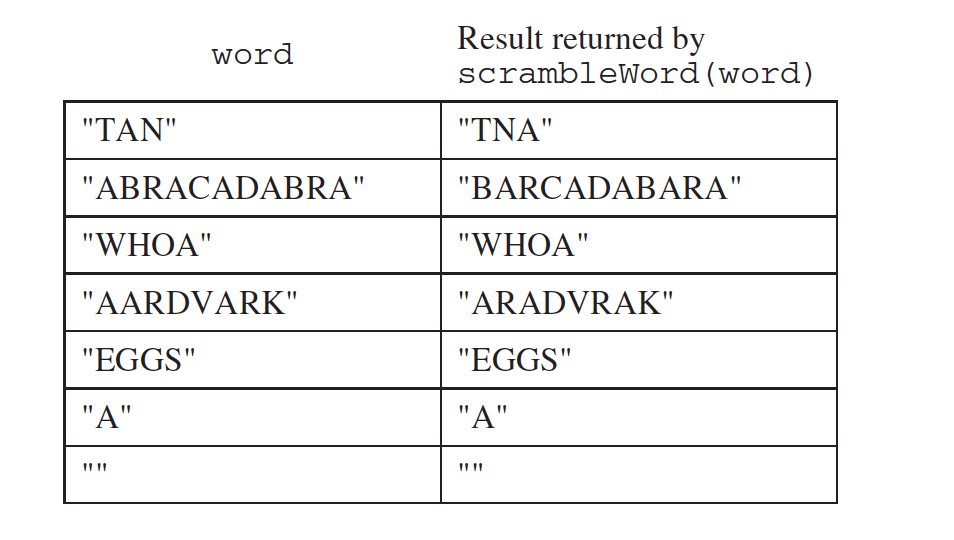
* + The main method is *in* the ScrambledStrings class, invoke it the same way you invoked other main methods.

**Question 1.** This question involves reasoning about strings made up of uppercase letters. You will implement two related methods that appear in the same class (not shown). The first method takes a single string parameter and returns a scrambled version of that string. The second method takes a list of strings and modifies the list by scrambling each entry in the list. Any entry that cannot be scrambled is removed from the list.

**Part a.** Write the method *scrambleWord*, which takes a given word and returns a string that contains a scrambled version of the word according to the following rules.

* The scrambling process begins at the first letter of the word and continues from left to right.
* If two consecutive letters consist of an “A” followed by a letter that is not an “A”, then the two letters are swapped in the resulting string.
* Once the letters in two adjacent positions have been swapped, neither of those two positions can be involved in a future swap.

The following table shows several examples of words and their scrambled versions.

[](http://interactivepython.org/runestone/static/JavaReview/_images/scrambleA.png)

**public** **class** **ScrambledStrings**

{

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Part (a) \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

*/\*\* Scrambles a given word.*

*\* @param word the word to be scrambled*

*\* @return the scrambled word (possibly equal to word)*

*\* Precondition: word is either an empty string or*

*\* contains only uppercase letters.*

*\* Postcondition: the string returned was created*

*\* from word as follows:*

*\* - the word was scrambled, beginning at the*

*\* first letter and continuing from left to right*

*\* - two consecutive letters consisting of "A"*

*\* followed by a letter that was not "A" were*

*\* swapped*

*\* - letters were swapped at most once*

*\*/*

**public** **static** String scrambleWord(String word)

{

*/\* to be implemented in part a \*/*

}

}