CPS 305 Project Four: Word Filling

Your code in `WordFiller.java` must be your own work. You are allowed to discuss the problem and solution ideas with each other on a purely theoretical level as long as **no actual code is shown to other students or changes hands.** The project submissions are checked automatically for similarity and plagiarism.

(**UPDATE NOVEMBER 8:**) However, you are allowed to freely copy and modify any code that is **directly linked** to on the course web page, or on the page Java Algorithms and Clients. No matter how much you copy and modify, it will not be considered plagiarism. You may not copy code from anywhere else, nor share copied code with other students.

The Problem To Solve

The fourth project for the course CPS 305 Fall 2018 is a very simple but still fun and interesting combinatorial problem of stringology made up by prof. Kokkarinen during one nice summer walk. The problem again concerns the words given in `sgb-words.txt` that contains 5757 five-letter English words, sorted by their frequency of actual use. (Same as the other standard datasets in Stanford Graphbase, this wordlist is set in stone by Donald Knuth, and is not for us mortals to modify in any way.)

Given a pattern that is guaranteed to contain only lowercase letters and asterisks, for example

```
l*n***f*is**rd*c*l*c**wia**b**y*e*p*uc*n*s*e**s*l*ps**a**s*l
```

your task is to construct a new string of the exact same length with the letters of the pattern in the same positions, and with each asterisk replaced by any letter of your choice to **maximize the total number of positions that contain some five-letter word**. For example, one possible answer

```
linertfaishardecalackswiaamberyleapsucanesternsclapsesarisel
```

would score 18 points for the 18 words (`liner`, `inert`, `shard`, `decal`, `alack`, `lacks`, `amber`, `beryl`, `leaps`, `canes`, `ester`, `stern`, `terns`, `claps`, `lapse`, `apses`, `saris`, `arise`) inside it.

These generated words may overlap, which makes this problem non-trivial and interesting as you try to fill in the letters to optimally them combine into words with their surroundings both to left and right.

Even though this does not happen in the above example, the same word may be used twice in different positions, and will receive a point each time.

Automated Tester

The instructors provide an automated tester `WordFillerTest.java` that will be used to check and grade the project. Used on the command line, it has the format

```
java WordFillerTest SEED LENGTH ROUNDS VERBOSE
```

where `SEED` is the seed given to the pseudorandom number generator that determines the patterns, `LENGTH` is the length of the patterns, `ROUNDS` is the number of random patterns to produce, and `VERBOSE` determines whether the tester prints out everything or just the final score line. An example run that consists of three rounds of patterns of the length 50 might look like the following:

```
matryximac:Java 305 ilkkakokkarinen$ java WordFillerTest 123456789 50 3 true
Read 5757 words from <sgb-words.txt>.

Seed  123456789: [na**sss*ie*p*re**g*o*o****oac*n*a*c**lu*t*r*oe*ti*]
Result returned in 1080 ms.
Returned result: [naapssstiesparedoggocoversoacannaoctalustsrdoestia]
pssst sties spare pared doggo cover overs verso canna octal talus lusts doest

Seed  123456790: [o*n*d**b***n*o**d*aa*t*e*h*i*a***bi*a*u*r**o**ac*s]
Result returned in 735 ms.
Returned result: [oanodesbacondomediaaateethcigardebitaburrskoalacks]
anode nodes bacon condo domed media teeth cigar garde debit burrs skoal koala alack lacks

Seed  123456791: [a**uo*lfr*hd*a*sa**e*no*aa**d*****t*i*e**po*ns***n]
Result returned in 5052 ms.
Returned result: [agluonlfrahdbalsablednoaaahydrovertricerspoonsalon]
gluon balsa sable abled hydro drove rover overt trice ricer icers spoon salon
41 6867 123456789 Kokkarinen, Ilkka
```

The last line of the tester output prints the total score (higher is better) and the measured total running time of your methods in milliseconds (lower is better), followed by your student information. If the command line option `VERBOSE` is set to `false`, the tester prints only this last line, but during your development stage, you will surely want to examine these verbose outputs. (Of course you can print your own additional debugging outputs during the development stage, but please remember to comment all those out before submission.)

Grading

All submissions will be tested in the same computer environment using the same secret `SEED` (to be decided by the grading TA's) and `LENGTH` of 50 for a total of ten `ROUNDS`. The submissions are sorted by their total scores, breaking ties between equal scores by the measured running times. The project marks between one and ten points are awarded based on this ranking. (The threshold for scoring five points is beating a simple Mickey Mouse solution concocted by the instructors for this purpose, to represent the minimum level of competency expected in this problem.) Furthermore, so that there is nothing flukey about the final winner, the top finishers are tested with a different `SEED` with `LENGTH` of 100 for a fifty `ROUNDS`, to shine the harsh daylight on the tiniest differences in the logic and speed of these top notch submissions.

Project submissions that crash or return an illegal answer will automatically receive a zero mark for this project. Furthermore, the testing has a total time limit of **one minute for these ten rounds**. Code whose execution does not terminate by that time will also receive a zero mark.

To allow the grading TA's to write Unix shell scripts to automate this testing, **your methods absolutely may not print anything on the console**. Any project submission that prints even one character on the console will automatically receive a zero mark.

Before submitting it, please have your code run overnight for at least a thousand rounds with `VERBOSE` set to `false`. If only the single result line and nothing else is there printed on your console when you wake up in the morning, you know that your code is not printing anything or crashing, and should be ready for submission.

As an added incentive and reward, for whatever it is worth, prof. Kokkarinen will write a letter of recommendation to the student whose project submission is ranked the first. The winner is also entitled to refer to him- or herself with the title of "The Fastest Gun South of Mississauga" for the duration of the Fall 2018 term.

Required Methods

Your submission must consist of exactly one source code file named `WordFiller.java` and no other files. It is acceptable to define nested classes inside this class. Your code is freely allowed to use all data structures and algorithms available in the Java 8 standard library so that you don't need to reinvent any of these wheels. Any attempt to interfere with the behaviour and results of the automated tester is considered cheating and will be punished by the forfeiture of all project marks in this course.

Your class must have the following exact methods so that the tester can compile and run:

```
public static String getAuthorName()
```

Returns your name in the format `"Lastname, Firstname"` exactly as it appears on RAMMS.

```
public static String getRyersonID()
```

Returns your Ryerson student ID.

```
public static void processWordList(List<String> words)
```

This method will be called exactly once by the automated tester in the beginning, giving you the list of legal `words` used in this problem. This method can do whatever preprocessing and data structure initialization you want to speed up the actual problem solving later. The time spent executing this method does not count towards the measured running time, but it does count towards the one minute total time limit.

Every word in `sgb-words.txt` consists of **exactly five letters** and can contain **only the 26 English lowercase letters** from `a` to `z`. This guarantee will hopefully simplify your preprocessing logic and the design of your data structures.

```
public static String fillPattern(String pattern)
```

The heart and soul of this project. Given the `pattern` of lowercase letters and asterisks, create and return the answer string in which you try to include as many words as possible. Do something intelligent here. As with all programming, start out simple to get it work correctly, and then think up gradual improvements and optimizations until the submission deadline to make this method both find better solutions and find them faster.

Side contest: Maximize a word chain

Date	Student	Words	Solution
Nov 8	Tyler Blanchard	17	`"routerodevillamassetheresterns"`

As a fun little sideshow, we are also running the following side contest whose winner will receive an extra five points to their total course grade. As a symbolic prize, the winner also earns the right to refer to him- or herself as "Dynamite Baby" for the remainder of the Fall 2018 term. We instructors don't know the correct answer, but again, it is trivial to check whether your submitted solution works. (It is almost as if there existed an entire class of computational problems that are simple enough to describe to a ten year old, and yet exponentially difficult to find the needle hiding in the massive haystack, even though any individual long and narrow thing can be trivially checked for its metallic sharpness...)

The contest: Find a string of **thirty characters** (we changed this, since 100 seems a bit excessive in retrospect) that contains as many different five-letter words from `sgb-words.txt` as possible.

Unlike in the main problem above, **repetitions of the same word are now counted only once**, to ensure that some trivial 5-cycle cannot be repeatedly used to fill up the entire string up to arbitrary lengths. Theoretically, this string could still contain up to 26 different words, but what is the highest possible count reachable with this particular list of 5757 English words?

The instructors will display on this page the current leading submission. If you find a string that contains more words than the current leader, submit it to `ikokkari@ryerson.ca` to become the new current leader. If multiple students submit working solutions that contain the same number of words, the student whose email arrives first in the inbox becomes the current leader. To topple the current leader, the submitted solution must contain at least one more word than the current leader.

You don't have to "show your work" at the time of submission since the submissions are self-verifying, but to receive the five points, the winner has to prepare a short document explaining how the solution was reached. We will then post this document on D2L for others to study and learn from.