CPS 305 Project Three: Packing Words in Bins

Your code in `WordPacking.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 third project for the course CPS 305 Fall 2018 is a combinatorial problem concerning 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 list of suitably randomly chosen words from this dataset, for example

```
avoid, spasm, these, prick, shunt, endow, degum, hapax, umbra, jacks, russe,
areal, modes, cinch, regal, bleak, spell, rille, glyph, jiffy, abaft, tangy,
coops, scant, cloth, hying, screw, props, grown, bravo, nicer, facie, weeny,
lords, exude, buffa, kudzu, molar, redox, elude, lacer, beaut, skiff, adman,
naked, unhip, prior, etext, gibes, rumor
```

your task is to pack these words into separate "bins" under the constraint that **no two words in the same bin are allowed to have the same letter in the same position anywhere in them**. For example, the words `frump` and `gouts` cannot go in the same bin, since both have the same letter `u` in the middle position. On the other hand, two anagrams `angle` and `glean` are made of the exact same five letters, but none of these letters are in the same position in both words, so these words can be placed into the same bin. (In more computer science-y terms, two five-letter words can be placed into the same bin if their Hamming distance is exactly five.)

Under this constraint, the previous fifty words can be packed into six separate bins by using the following arrangement, one of the exponentially many equally valid such arrangements possible.

```
Bin 1: [avoid, these, prick, hapax, umbra, modes, glyph, jiffy, scant, rumor]
Bin 2: [jacks, screw, bravo, weeny, exude, kudzu, adman, unhip]
Bin 3: [shunt, endow, degum, russe, bleak, tangy, grown, nicer, lords]
Bin 4: [spasm, regal, cloth, facie, buffa, prior, etext, gibes]
Bin 5: [areal, rille, coops, lacer, beaut, skiff]
Bin 6: [cinch, spell, abaft, hying, props, molar, redox, elude, naked]
```

Only the total number of bins used to achieve the packing arrangement matters; how evenly the words are distributed into these bins is irrelevant. (In practice, minimizing the total number of bins seems to tend to create roughly equally sized bins.)

An equivalent, but again more computer science-y, way to phrase this problem would be to ask for a **minimum vertex colouring** in the SGB word graph in which two word vertices are connected by an edge if and only if those words share the same letter in at least one position. Bins, colours, to-may-to, to-mah-to. Those are all mere words anyway, and the problem itself and the universe of "physics of the abstract" that encompasses it do not care one jot or tittle which particular flappings of the vocal cords we meatsacks happen to use to talk about it.

Automated Tester

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

```
java WordPackingTest SEED SIZE ROUNDS VERBOSE
```

where `SEED` is the seed given to the pseudorandom number generator that determines the lists of words to pack into bins, `SIZE` is the number of words in each problem instance, `ROUNDS` is the number of random lists to produce for packing, and `VERBOSE` determines whether the tester prints out everything or only the final score line. An example run that consists of three rounds of lists that contain 50 words each might look like the following. (**Update Nov 3**: My previous model solution had an optimization bug that I did not realize was there until I solved this problem today using another, completely different algorithm. The output below is from the second solution that uses 20 bins whereas the previous buggy solution used 22. After fixing the bug, both solutions use 20 bins. Unless I have some other bug in my code, I believe this 20-bin solution is now optimal.)

```
matryximac:Java 305 ilkkakokkarinen$ java WordPackingTest 777777 50 3 true
Read 5757 words from <sgb-words.txt>.

Seed 777777: [nutsy, deter, dicot, gulch, adman, junta, skint, phone, which, marks, ethyl,
owned, mylar, unfit, scrub, skunk, myths, slots, cedar, lisle, undue, hinds, estop, dream,
jibed, impel, patch, sibyl, quark, combo, probe, ennui, polka, gamin, daddy, atone, views,
torah, asked, shush, whirr, cools, hoard, whang, ftped, harpy, ocean, rutty, found, runty]
Solution found in 81 ms.
Bin 1: [deter, adman, junta, which, marks, sibyl, probe, found]
Bin 2: [skint, mylar, undue, impel, patch, views, hoard, runty]
Bin 3: [dicot, owned, myths, cedar, quark, polka, atone, shush, harpy]
Bin 4: [phone, ethyl, scrub, hinds, dream, combo, asked, rutty]
Bin 5: [nutsy, slots, lisle, ennui, gamin, torah, whang, ftped]
Bin 6: [gulch, unfit, skunk, estop, jibed, daddy, whirr, cools, ocean]

Seed 777778: [hexed, recap, carry, rodeo, emote, litho, pager, trick, crown, faint, clink,
typos, chino, waged, trawl, bowls, jihad, petit, kelly, axman, sloop, dogma, elate, sutra,
vitas, dials, raspy, gypsy, demon, unapt, lucky, cruse, snark, wurst, ghoul, sweep, tenth,
noddy, ghoti, mound, bride, upped, range, minas, nymph, swede, forth, wrong, sigma, flits]
Solution found in 2966 ms.
Bin 1: [recap, carry, typos, unapt, ghoti, mound, bride, sigma]
Bin 2: [emote, pager, clink, trawl, bowls, jihad, demon, wurst]
Bin 3: [rodeo, crown, faint, gypsy, tenth, swede]
Bin 4: [waged, petit, axman, sloop, dials, lucky, cruse, forth]
Bin 5: [chino, elate, sutra, raspy, upped, minas]
Bin 6: [hexed, litho, snark, noddy, range, wrong, flits]
Bin 7: [trick, kelly, dogma, vitas, ghoul, sweep, nymph]

Seed 777779: [stash, decaf, onion, rehab, goest, armor, spark, ixnay, jives, hippy, algin,
rived, dunno, radix, swoon, noose, arise, hadda, bream, crack, klieg, agony, furls, bombe,
admit, hullo, close, rapid, bones, seams, igloo, rawly, cause, whelp, loads, hymns, dicot,
fiche, avail, chord, spoil, cruft, yerba, fauna, stung, clunk, quick, token, umber, hertz]
Solution found in 269 ms.
Bin 1: [armor, spark, hullo, close, rapid, bones]
Bin 2: [rehab, algin, noose, hadda, crack, whelp, dicot, stung]
Bin 3: [decaf, hippy, arise, furls, igloo, spoil, clunk, token]
Bin 4: [rived, bream, agony, seams, cause, quick]
Bin 5: [ixnay, admit, loads, chord, fauna, umber]
Bin 6: [goest, swoon, klieg, rawly, hymns, fiche, avail, yerba]
Bin 7: [stash, onion, jives, dunno, radix, bombe, cruft, hertz]
20 3316 123456789 Kokkarinen, Ilkka
```

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

To keep this problem interesting, note that the random selection of the wordlist done inside the automated tester is artificially biased so that it intentionally avoids words that contain letters in positions that have already been used too many times in the previously selected words.

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 `SIZE` 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 ten finishers are tested with a different `SEED` with `SIZE` of 100 for one thousand `ROUNDS`, to bring to light every little difference in the logic and speed of these top submissions.

Project submissions that crash or return an illegal answer 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, 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 when you wake up in the morning, you know that your code is not printing anything, getting stuck in an infinite loop, or crashing.

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 North of Mississauga" for the duration of the Fall 2018 term.

Required Methods

Your submission must consist of exactly one source code file named `WordPacking.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 List<List<String>> wordPack(List<String> words)
```

The heart and soul of this project. Given the list of `words`, create and return a list whose each element is a list of strings that contains the words that you choose to place inside that particular bin. Every word in `words` must occur exactly once in exactly one of these lists, and no two words in the same list may contain the same letter in the same position anywhere. You can freely choose the exact subtype of the list and the lists that it contains, as long as they are subtypes of `List<E>`.

Side Contest: Maximize One Bin

Date	Student(s)	Size	Solution
Nov 7	Tyler Blanchard	17	`[pffft, ethyl, bally, vents, whose, litho, dweeb, using, abuzz, mujik, cramp, zombi, scram, incur, glyph, oxbow, hydra]`
Nov 5	Tyler Blanchard	16	`[lambs, flank, trice, cowed, input, bingo, nervy, excon, abuzz, umbra, sylph, dwell, pshaw, optic, whomp, mufti]`
Oct 29	Deep Patel, Calvin Yap	15	`[after, backs, cease, diddy, eclat, fjord, glitz, hokum, idyll, jumbo, lynch, omega, rhumb, sprig, unbox]` (plus 25 other combos of that same size)

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 call him- or herself "The Soldering Iron of Justice" 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 to describe and yet exponentially complex to solve by finding 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 the largest possible subset of words in `sgb-words.txt` that **can be packed into a single bin.** Theoretically this subset could contain up to 26 words (every word consumes one of the 26 possible letters from each of its five positions), but what is the largest such subset that actually exists within this particular list of 5757 English words?

The instructors will display on this page the current leading submission. If you find a subset 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 my 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 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 post this document on D2L for others to study and learn from.