Java Search Project: Packing Words in Bins

The Problem To Solve

To serve as a standard dataset for his work first with [Stanford Graphbase]() and then with Volume 4 on combinatorial algorithms, Donald Knuth compiled and set in stone a collection of 5757 five-letter English words, given sorted by their frequency of actual use in the text file `sgb-words.txt`. In this spirit, students may use any and all combinatorial search and optimization techniques in their mental arsenal to solve the following fun little combinatorial problem.

Given a set of words, arrange them individually into as few "bins" as possible, under the hard constraint that **no two words in the same bin may have the same letter in any position**. For example, the words `frump` and `gouts` can't go into the same bin, since they both have the same letter `u` in the middle position. On the other hand, the two anagrams `angle` and `glean` have the exact same five letters, but since none of these letters are in the same position in both words, these words go painlessly into the same bin.

For example, the following randomly chosen subset of sixty words

```
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
```

can be arranged into six separate bins as given below, one of the many equally valid possible arrangements. These six bins turn out to be as good as it gets; Whichever way you twist and turn and shout and pout, these words won't go into five bins without at least one constraint violation unavoidably taking place somewhere.

```
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]
```

(Yet another equivalent way to phrase this problem is to ask for a **minimum vertex colouring** in the **word graph** with words as nodes, so that two word nodes are connected by an edge if and only if their **Hamming distance** is less than five. Bins, colours, to-may-to, to-mah-to. Those are mere words anyway, as 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 us smelly meatsacks happen to use when talking to ourselves about it.)

Automated Tester

As usual, an automated tester WordPackingTest.java will be used to grade this project. When used on the command line, it has the format

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

where SEED is the seed for the tester's pseudorandom number generator, SIZE is the number of words in each problem instance, ROUNDS is the number of rounds to play this game, 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 each contain 50 words should resemble the following.

```
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 shows the total score (lower is better) followed by the total measured running time of the entire run, given 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. However, during your development, you will surely want to examine these verbose outputs. (Of course you may also print your own debugging outputs during development, just remember to comment them all out before submission.)

To make this problem more fun, note that the construction of the random wordlist is done with the "thumb on the scale" to avoid words that contain letters in positions that have already been used too many times in the previously selected words. In general, problems whose constraints are so hard that they essentially force you directly to the only possible solution are usually not that interesting. Same as with other NP-complete combinatorial search problems, there is a "sweet spot" for the number of random constraints that allows its chaos of latent possibilities to burst out and genuinely flower.

Grading

Student submissions will be tested by the instructor under the same computer environment with the exact same secret `SEED`, using the `SIZE` of 50 for a total of ten `ROUNDS`.

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. 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 always, silence is golden. **Your methods may not print anything during their execution**. Any project submission that prints even a single character on the console will automatically receive a zero mark.

Any attempt to interfere with the behaviour and results of the automated tester in any way, even as a prank, is plain and simple cheating, and will be punished by the forfeiture of all project marks for this course.

Required Methods

Your submission must consist of **exactly one Java 8 source code file** named `WordPacking.java` and no other files. You may define nested classes inside this class. You may use any and all data structures and algorithms provided in the Java 8 standard library. This way you don't need to reinvent any of those rusty old wheels, but can concentrate full time on the logic of your search and optimization.

Your class **must have the following exact methods** to allow the tester to compile and run:

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

Returns your name in the format "`Lastname, Firstname`".

```
public static String getStudentID()
```

Returns your student ID. Please do not include any whitespace characters in your ID.

```
public static List<List<String>> wordPack(List<String> words)
```

This method is the heart and soul of this project. Given the list of `words`, create and return a list of bins whose each element is a list of strings, the contents of that particular bin. The automated tester enforces that every word in `words` appears in exactly one bin, and that no two words in the same bin contain the same letter in the same position.

You can freely choose the exact subtype of the returned list object and the lists inside it, as long as they are subtypes of `List<String>`. Most likely, simple `ArrayList<String>` is best.

Historical Side Contest: Maximize One Bin

This instructor first created and used this problem during his stint of daytime teaching back in the Fall 2018 term. At that time, he also realized that this setup is pregnant with an interesting sideshow tent problem screaming to burst out; If you get to choose your words freely from the 5757 words included in the file `sgb-words.txt`, how many words could you fit into a single bin?

Interested students were offered this optional side quest with the reward of ceremonial title of "The Soldering Iron of Justice" for the duration of that term for the student or pair of students who first finds the largest subset of words that together fit into a single bin. Theoretically this subset could contain up to 26 words, since every word consumes one of the 26 possible letters from each of its five positions. However, same as all other real world datasets, Knuth's wordlist rarely contains the items that we really hope it to have, so in practice we fall far short of this ideal.

The following table shows the history of students improving the best solution known at the time.

Date	Student(s)	Size	Solution
Nov 23	Ryan Chan Shum Hang	18	`[ethyl, oxbow, injun, affix, nymph, kudzu, uvula, thwap, pssst, gizmo, skiff, flyby, wrong, bocci, hertz, dweeb, valve, czars]`
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]`

The solution with 18 words seems to be optimal. Anyone who comes up with a solution with 19 words instantly becomes the permanent holder of the ceremonial title of “The Soldering Iron of Heavens Justice”, at least as far as this instructor is concerned.

Motivated students might also repeat this analysis with wordlists of other languages to compare the results. For example, how large a bin could one create filled to its brim with five-letter words of Spanish or French? (Some cunning linguist with an actual expertise in these matters probably ought to decide whether various accented versions of certain letters are considered the same character.)