Gaming fan's nadir

Two expressions are anagrams of one another if, ignoring capitalisation, punctuation and whitespace they contain the same characters. For instance `Finding Anagrams` and `Gaming fan's nadir` are anagrams. Utilities such as I rearrangement servant can help find anagrams which arise most commonly in cryptic crossword puzzles but also in other sorts of wordplay. The intent of this étude is to ask you to reproduce some of the functionality of such utilities.

Task

Write a program that, using a user supplied dictionary, finds all anagrams of a given expression using at most a certain number of words from the dictionary. Assuming Java as the programming language the program (say `Anagrams`) should be invoked as follows:

```
java Anagrams "**Finding Anagrams!**" 3 < dictionary.txt
```

This would output to `stdout` all anagrams of `**Finding Anagrams!**` consisting of at most three words from the supplied dictionary. The dictionary will contain up to 100,000 strings consisting only of lower case characters from `a` through `z`, one per line. Note that the input string may well require some modification (as in this example - the output would be the same if the input were `findinganagrams`).

The anagrams you find must be output to `stdout` in the following order:

- Individual anagrams should be listed on a single line in descending order of word length, with words of equal length in alphabetical order.
- Anagrams using fewer words precede ones using more words.
- If two anagrams use the same number of words, they should first be compared by word length – if the longest words differ in length, then the one with a longer first word should be listed first. If the longest words are equal in length but the second longest words differ in length then the one with a longer second longest word should be listed first. And so on.
- If two anagrams have the same number of words, and corresponding words have the same length, then they should be listed in alphabetical order.

(Pair 2)