Anagrams Application flow and Design justification

Assumptions:

- Assuming that list of anagrams should all be distinct thus HashSet is used.
- Words containing characters other than alphabets (A-Z and a-z) will get ignored.
- Spaces or empty lines will be ignored while reading from input file.

- readAllLines is used to read input text file. This function is used to read fairly small files but if we want to read very large text files we should consider using bufferReader.

- We certainly don't want to store empty lines or lines with spaces only in hashMap thus regular expression check for the validity of the word.

- Tries were considered before using hashMap in creating a dictionary since tries can be very effective when we have limited alphabet characters and high redundancy of reappearing those words or alphabets. But when it comes to searching, tries are good for prefix based searching where two or more words have similar initial characters. But for anagrams searching a trie could be inefficient since the order of characters in anagram doesn't matter.

- Thus, Hashmap was used. Every word from input file is first sorted and the sorted word serve as key of Hashmap. By sorting words which are anagrams of each other they become same. For example, 'act' and 'cat' when sorted it becomes 'act'. 'act' will serve as key to the hashMap. It is then checked if there exist any value against this key. If yes, then new value is added to the returned hashSet otherwise a new hashSet is created and value is added in the hashSet.

- Many other data structures can be used instead of hashSet to store the values/anagrams such as arrayList, LinkedList or another hashMap. But preference is given to hashSet because of two reasons. Firstly, if there exist duplicate words in the dictionary then they will be eliminated since hashSet cannot hold duplicate values and secondly hashSet have constant lookup time against key. In future if the requirement changes and we'd like to search if the dictionary contains word 'sale' whose anagram is 'seal' and we have our dictionary built and ready. We can pass the word and check if it exists, by first sorting 'sale' and then passing it in hashMap. HashMap returns value in form of set. In order to check if word 'seal' exist in set we can simply pass the word in hashSet and check the returned Boolean value. Whereas if we would have used any of list data structure we'd have to traverse through the returned results to find out.

- The time complexity of this application will be O(n) where n is number of valid words in input file.