## Dictionary

By: Abhijeet Mehrotra

Language used: Java 1.8
Compiled jar can be run by: java -jar dictionary.jar

**Assumptions:**

1. English dictionary with no special characters allowed.

2. Words:
   Allowed: abc, def
   Not allowed: ab**cd, *abc, abc*
   Note: spaces are trimmed in words. So, " abc " is reduced to abc

3. Only spaces allowed in definitions apart from alphabets.
   Allowed:"abc def"
   Leading and trailing spaces are trimmed.

4. 1:1 mapping between words and definitions.

5. All characters are converted to lowercase before usage.

**Design:**

1. The Dictionary is made up of Nodes.
2. There is an empty root node which in turn points to child nodes if they exist.
3. Every node level represents a character level in the dictionary.
4. Along the path, when a word (made by concatenating all the characters from the node to the root) exists, it's corresponding definition is stored in the node as a string.
5. Only the paths that are needed are maintained, cleanup is performed on deletion.

No imports are used as required. Except, java.util.ArrayList which is used simply to make a list of strings the prefix matches to and return them.

- The dictionary is exposed through the class Dictionary.

  **Functions in Dictionary:**

  1. boolean insertWord(String word, String defn)
     Returns:
     true: on success
     false: on failure
     Already existing word definition is overwritten.

2. ArrayList<String> prefixSearch(String word)
   Returns:
   List of strings: on success
   Empty list : on failure or no words found.

3. String getDefinition(String word)
   Returns:
   String: on success
   null: on failure or word not found.

4. boolean deleteWord(String word)
   Returns:
   true: on success
   false: on failure

I encapsulated errors of invalid words, nonexistent words etc. in true and false, because the dictionary is not persistent.

- It is tested in Test.java.
- Node.java contains the node definition.
- Utils.java contains the helper functions.