A picture containing logo

Description automatically generated

King Fahd University of Petroleum and Minerals

221-ICS 202 PROJECT

[Dictionary – AVL Tree Data Structure Implementation]

MOHAMMED ALMUBARAK – 202024880 -SEC#51

HUSSAIN ALKHALAIF – 202035120 - SEC#51



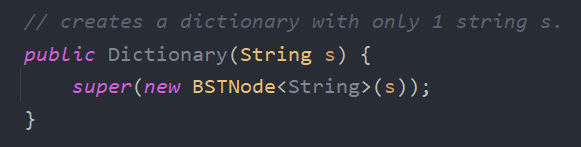
The dictionary holds a list of words (strings) to be used in a spell checker. The class is a subclass of the AVLTree class that also substitutes the generic T to a type String, since the dictionary only holds words (strings) and also to get rid of unnecessary casting and issues had we used a generic type T.

**1- Initialization**

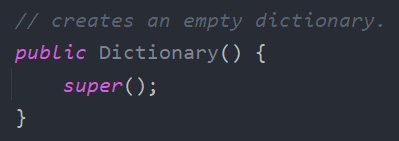
First of all, we created a dictionary class that extends AVLTree<String>, that can be initialized using three different constructors with different parameters:

1. a single string, [public Dictionary(String s)]
2. an empty dictionary, [public Dictionary( )]
3. a text file having strings, each on a new line, [public Dictionary(File f)]

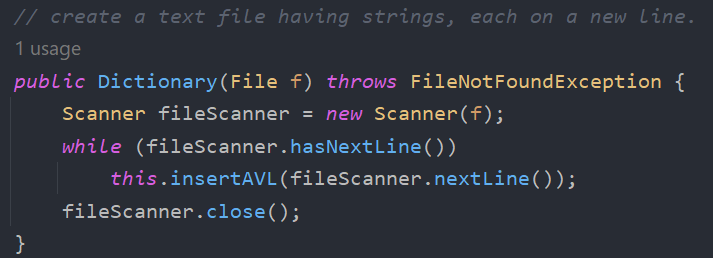
In the first constructor, we create a dictionary with only one string which would be the root in the actual AVL tree using the super keyword. This uses the original AVLTree constructor.



In the second constructor, we create an empty dictionary using the super keyword. This uses the original AVLTree constructor.

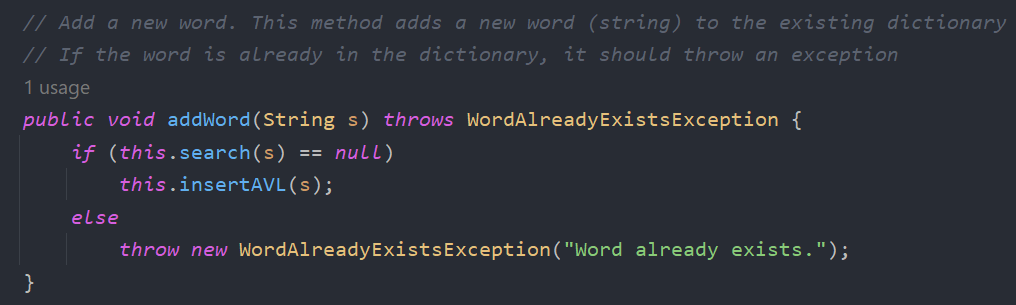


In the third constructor, we are reading a preformatted text file and inserting each word in it to the AVL tree. The method will throw an exception if the file does not exist or is inaccessible, which should be handled later on when using this constructor.



**2- Adding a new word**

This method adds a new word to the dictionary. It searches for a given word, if it does not exist, then it will add it to the tree using the super class’s insertAVL. Otherwise, if it already exists, it will throw an exception.



**3- Searching for a word**

This method of return type Boolean takes one string and searches for the given word. Then, if it’s found, the method will return true, and if it not found, it would return false.

For searching, we are using the search method from the BST class.

Text

Description automatically generated

**4- Removing a word**

This method takes one string to delete it from the dictionary. It first searches for the given word. If it’s found, it would delete the word from the AVL tree, and if it isn’t found, then it would throw an exception to be handled later.

For searching, we are using the search method from BST class, and to delete, we use deleteAVL from AVLTree class, which also handles re-balancing the tree if needed.

Text

Description automatically generated

**5- Finding similar words:**

This method takes one string to find similar words to it in the dictionary. It searches for words that have the same length or differ by one letter in the length. If such words are to be found, the method stores then in a stack, then pushes them in another stack to restore the correct order, and finally formats the output as a string and returns it. It uses a helper recursive method to start searching.

Text

Description automatically generated

**Helper findSimilar method: Part 1**

Text

Description automatically generated

In this part, we first recursively call the method to search in the left subtree, and then we check if the input word length is equal to the word stored in the current node. If lengths are equal, then we initialize a variable called differentLetter to count how many letter the two words are different from each other. We check each character and increment differentLetter as necessary. Upon finishing, if we had gotten differentLetter = 1, then we push the similar word to the stack.

**Helper findSimilar method: Part 2**

Text

Description automatically generated

If the previous condition (same length) is not true, then check if the input word length greater than the current node’s word by 1, then we initialize a variable called differentLetter to count how many letters the two words are different from each other. We check each character, and if we get a different character, we increment our differentLetter counter. When this process has finished, and get differentLetter = 1 or = 0 (which means there is no difference or the different character is the last character in the input word), then push the similar word to the stack.

**Helper findSimilar method: Part 3**

Text

Description automatically generated

If the previous condition is not true, then check if the current node’s length is greater than the input word by 1, then we initialize a variable called differentLetter to count how many letters the two words are different from each other. Again, we apply the same process: we check each character, and if we get a different character, then we increment our differentLetter counter. When this process has finished and we get differentLetter = 1 or = 0 (which means there is no difference or the different character is the last character in the input word), then push the similar word to the stack.

After all that, we then recursively the call the method to find similar words in the right subtree, too.

**6- Save the updated dictionary:**

After adding or removing words from the dictionary, we can save it as a new text file using this method. This method uses a modified version of in-order depth-first traversing.

Text

Description automatically generated

**Custom exceptions:**

* **Custom WordAlreadyExistsException:**

Text

Description automatically generated

* **Custom WordNotFoundException:**

Graphical user interface, text

Description automatically generated

**Test class:**

Text

Description automatically generated

Text

Description automatically generated

* **Catches in the test class:**

Text

Description automatically generated

1. **Initialization**

* **A text file having strings.**

Graphical user interface, text

Description automatically generated

* **A single string.**

Text

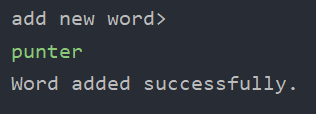
Description automatically generated

* **Empty.**

A screenshot of a computer

Description automatically generated with medium confidence

1. **Add new word**



Text

Description automatically generated

1. **Search for word**

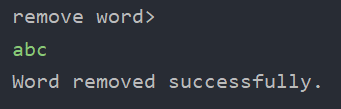
A screenshot of a computer

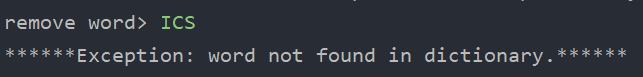
Description automatically generated with low confidence

Graphical user interface, text

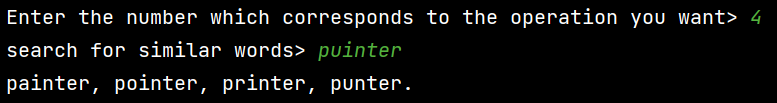
Description automatically generated

1. **Remove word**





**5) Find similar words**



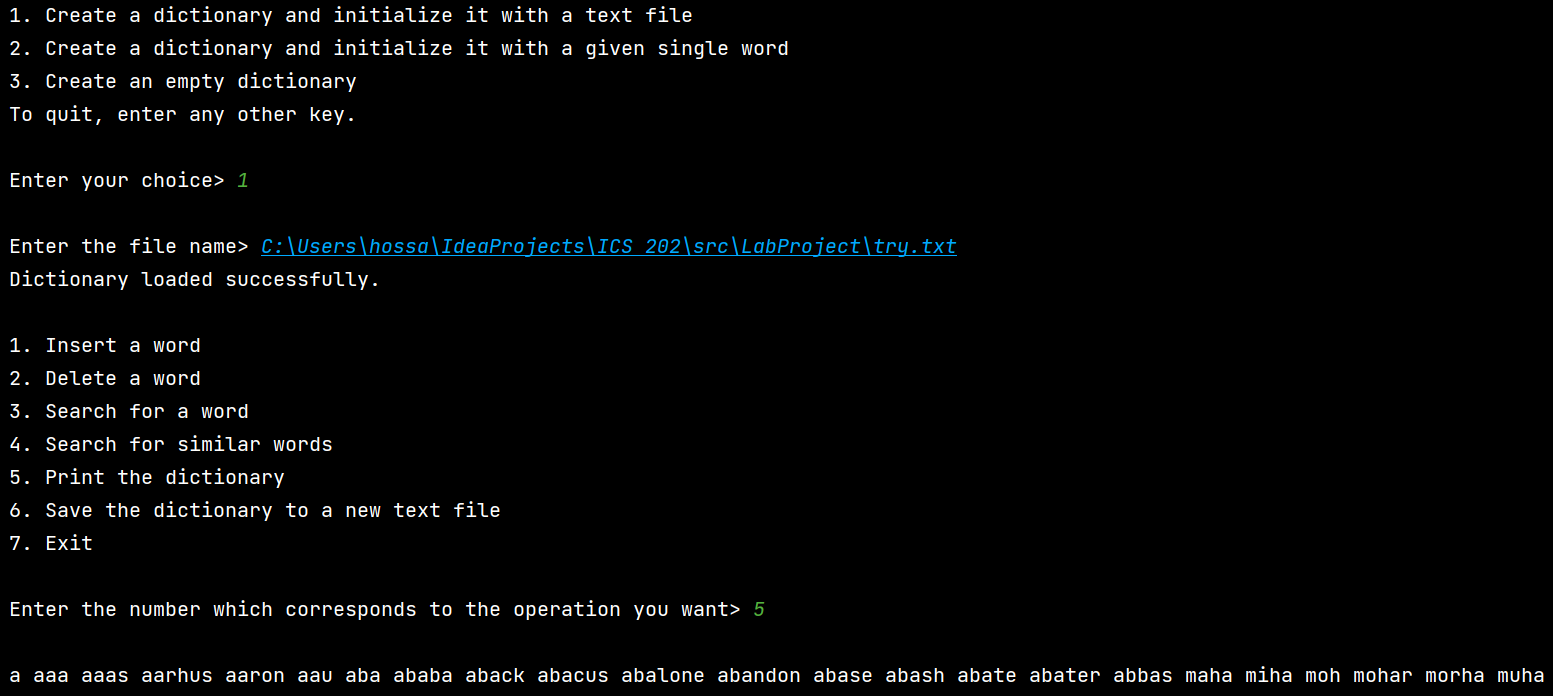
**6) Save file**

**Text

Description automatically generated**

Graphical user interface, text

Description automatically generated with medium confidence

**7) Printing the dictionary**