Names: Doyoung Kim, Rohan Tanna

UTEID: dk24338, rrt494

Section: 16185

EE422C

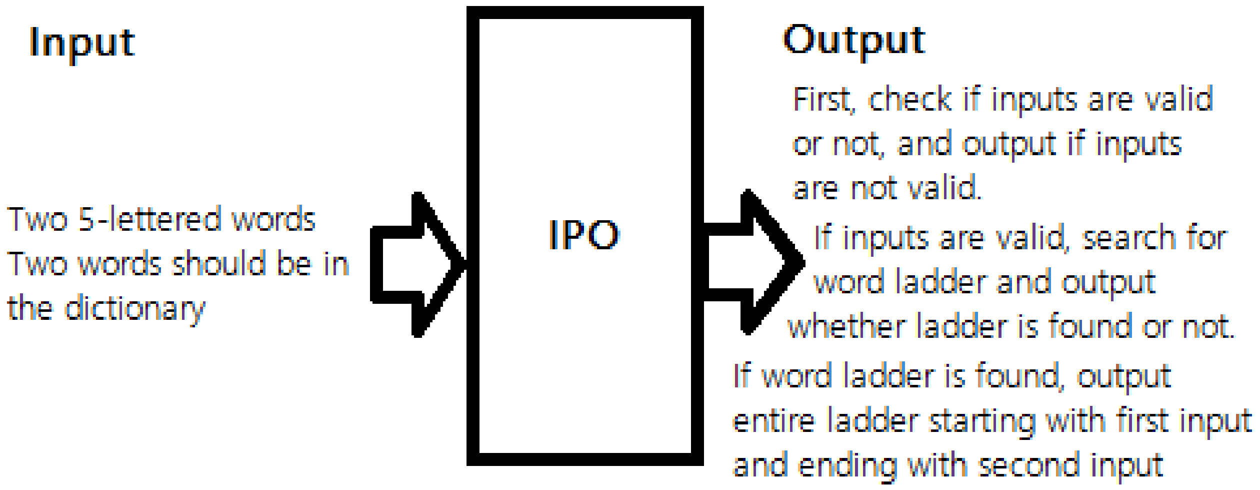
Assignment 4: Word Ladder

Analysis

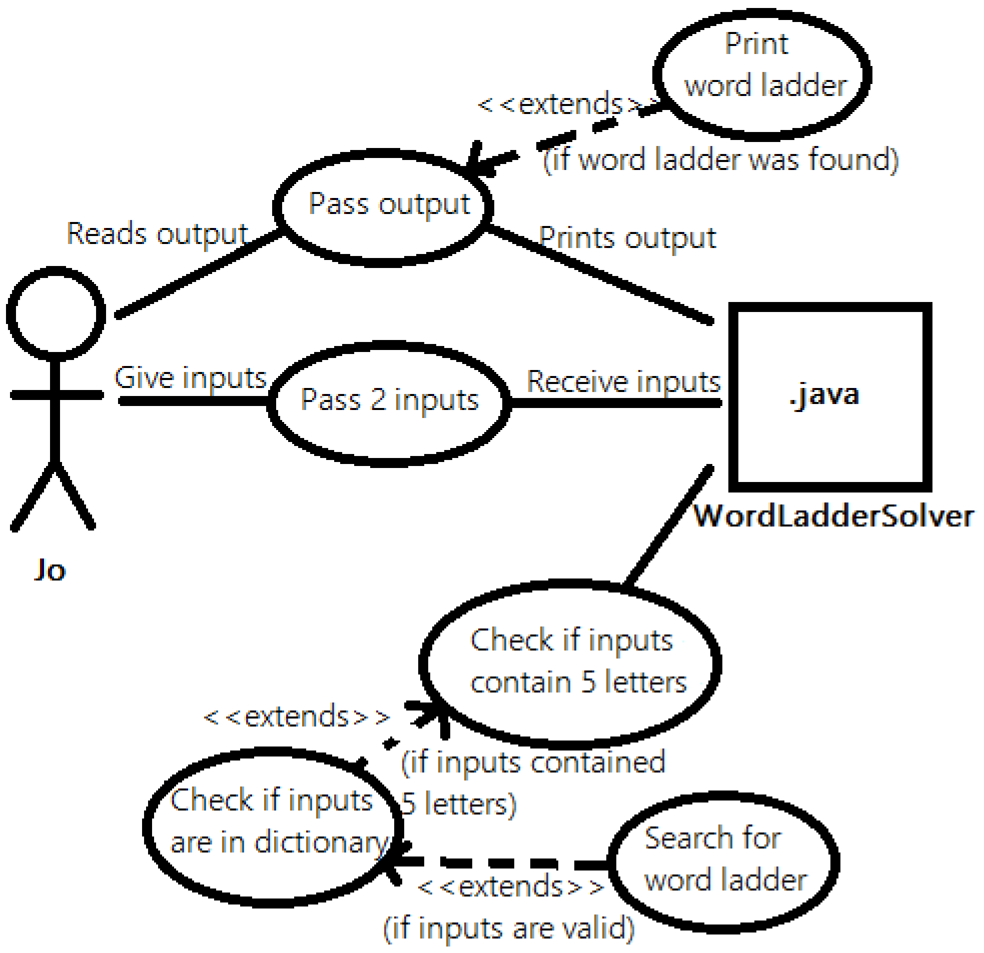
Implement WordLadderSolver class, which finds sequence of distinct English words such that any two consecutive words in the sequence is differ by changing one letter at a time. Solver method should take two inputs and one should be the start of the sequence and other should be the end of the sequence. Both inputs needs to be a 5-letter words that are in given dictionary and program should check if inputs are correct before starting the search. Note that it is possible to not find any word ladder from one end to the other.

Design

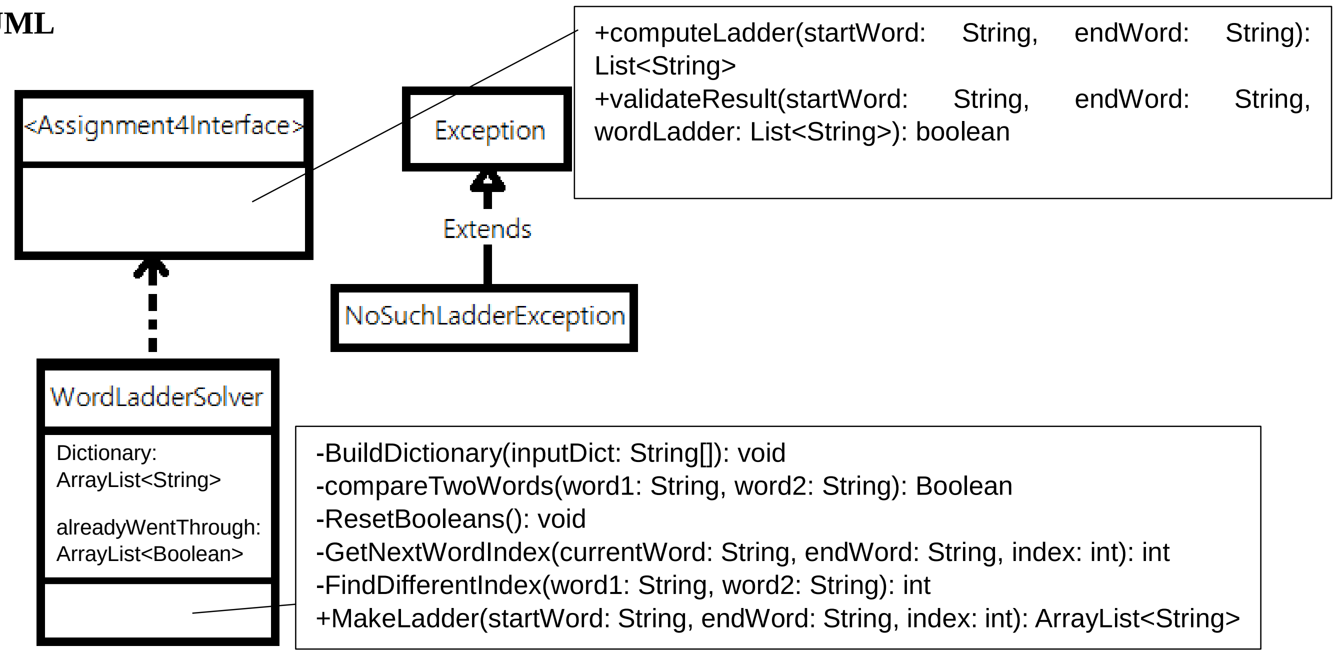
IPO


Input
Two 5-lettered words
Two words should be in the dictionary
IPO
Output
First, check if inputs are valid or not, and output if inputs are not valid.
If inputs are valid, search for word ladder and output whether ladder is found or not.
If word ladder is found, output entire ladder starting with first input and ending with second input

Use Case



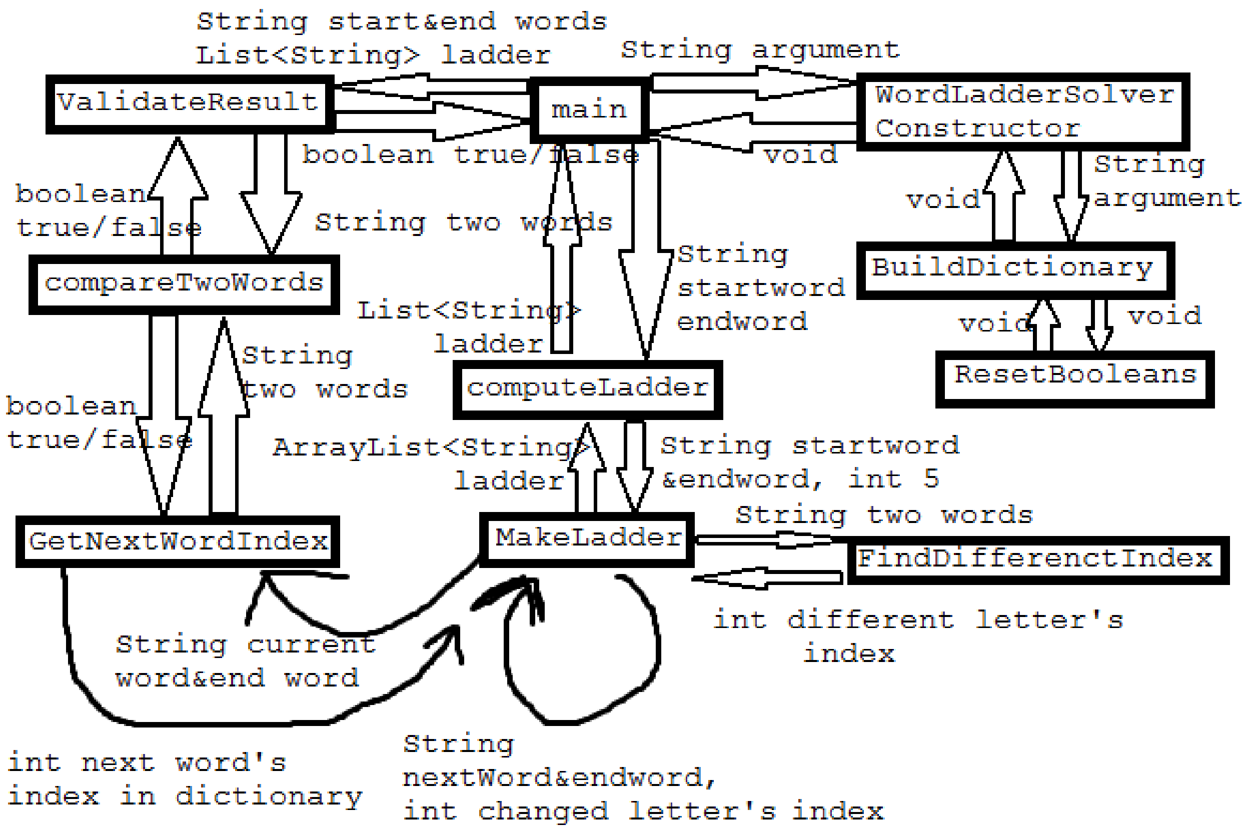
UML



+computeLadder(startWord: String, endWord: String): List<String>
+validateResult(startWord: String, endWord: String, wordLadder: List<String>): boolean

-BuildDictionary(inputDict: String[]): void
-compareTwoWords(word1: String, word2: String): Boolean
-ResetBooleans(): void
-GetNextWordIndex(currentWord: String, endWord: String, index: int): int
-FindDifferentIndex(word1: String, word2: String): int
+MakeLadder(startWord: String, endWord: String, index: int): ArrayList<String>

Functional Block Diagram



Pseudo Code

ComputeLadder

1. Check check if two input words are valid. Throw exception if one or more are invalid.
2. Check if two input words are same. Return List with of two words if they are same.
3. Make ladder
    A. Check if two words are different by only one letter. Return list if they are.
    B. Get list of words from dictionary that differ by one letter compared to current word.
    C. Get list containing word ladder by recursively starting from 'A' with one of word and end word.
        i. If word ladder does not contain end path, word ladder is not found, so try next adjacent word.
        ii. If word ladder contain end path, return the list.
4. Check if ladder is valid. (If there was path). Out accordingly and return list.

A paragraph describing the rationale behind your design. This would include:
a) How does your OOD reflect the interaction and behavior of the real-world objects that it models

Our design excludes possible infinite loop or non-smart laddering like change same letter index multiple times. When searching for path from one point to other, person would not go around in one loop forever. We designed so that our program will behave like human when one's given dictionary, starting word, and ending word to find word ladder.

b) What alternatives did you consider? What were the advantages/disadvantages of each alternative both from a programming perspective and a user perspective?

Using graph instead of arraylist of words.

It takes processing time and extra codes to create graph, but graph may boost the time taken for search.

c) What are some expansions or possible flexibilities that your design offers for future enhancements?

Dictionary can read word with any number of letter.

d) How does your design adhere to principles of good design: OOD, cohesion, coupling, info hiding,

Most of method used by public methods are private and variables in the class are private.