Names: Doyoung Kim, Rohan Tanna

UTEID: dk24338, rrt494

Section: 16185

EE422C

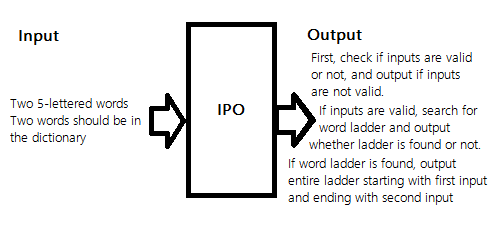
**Assignment 4: Word Ladder**

**Analysis**

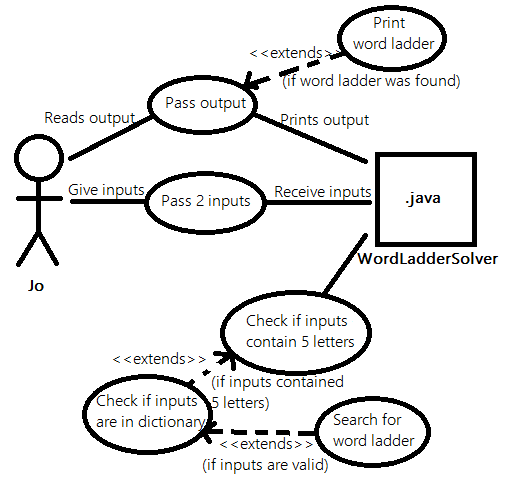
Implement WordLadderSolver class, which finds sequence of distinct English words such that any two consecutive words in the sequence is differ by changing one letter at a time. Solver method should take two inputs and one should be the start of the sequence and other should be the end of the sequence. 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**

****

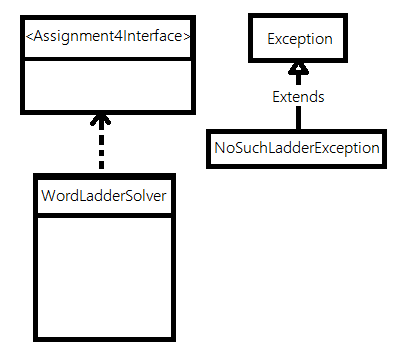
**Use Case**

****

**UML**

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

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

****

Dictionary: ArrayList<String>

alreadyWentThrough: ArrayList<Boolean>

-BuildDictionary(inputDict: String[]): void

-compareTwoWords(word1: String, word2: String): Boolean

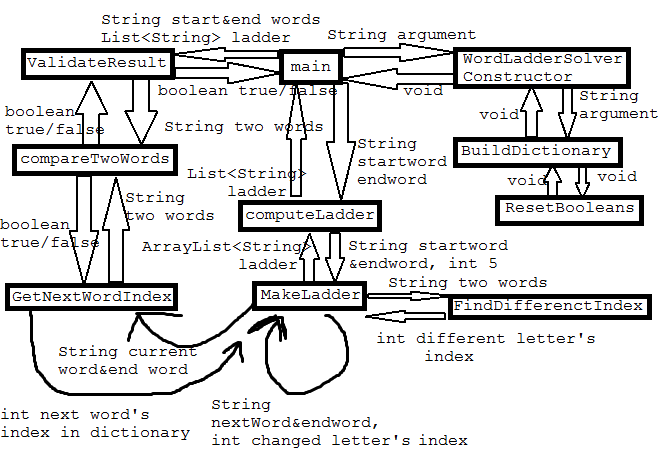
-ResetBooleans(): void

-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
   1. Check if two words are different by only one letter. Return list if they are.
   2. Get list of words from dictionary that differ by one letter compared to current word.
   3. Get list containing word ladder by recursively starting from ‘A’ with one of word and end word.
      1. If word ladder does not contain end path, word ladder is not found, so try next adjacent word.
      2. 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  
b) What alternatives did you consider? What were the advantages/disadvantages of each alternative both from a programming perspective and a user perspective?  
c) What are some expansions or possible flexibilities that your design offers for future enhancements?  
d) How does your design adhere to principles of good design: OOD, cohesion, coupling, info hiding,  
etc?**