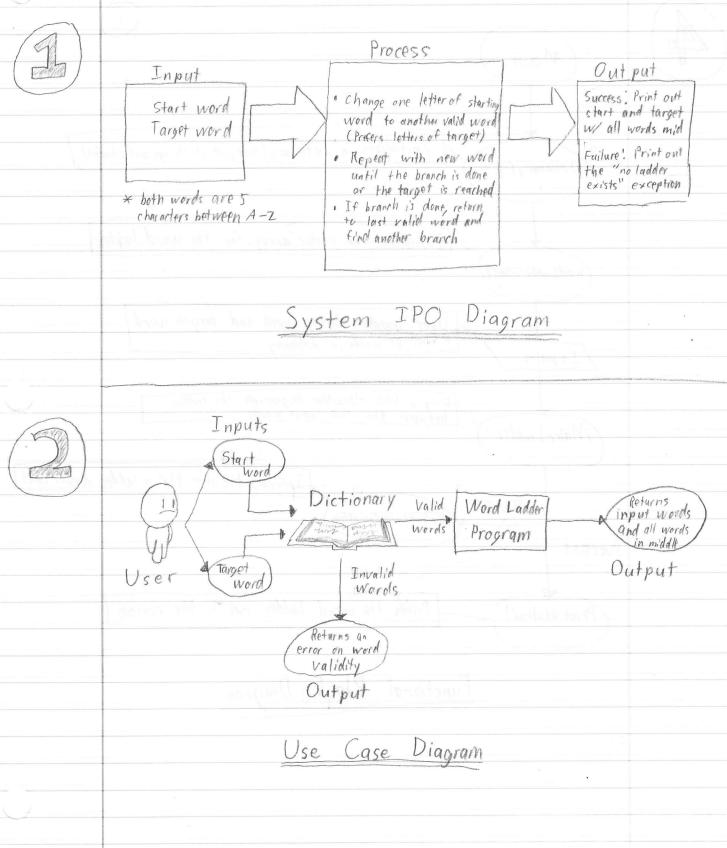
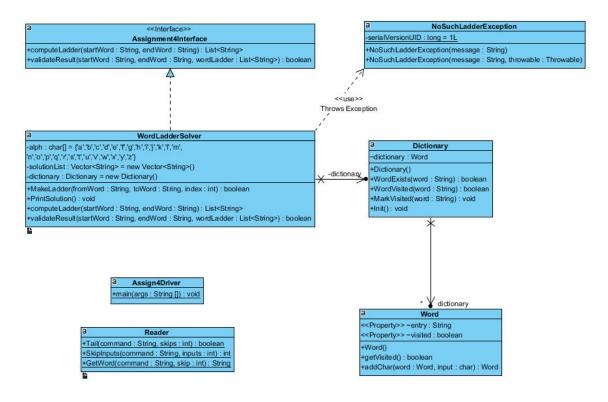
Assignment 4: Word Ladders

EF 422C



Design

UML Model



Driver Algorithm

- Read input File
- For every input do the following...
 - Parse StartString and EndString
 - Verify That StartString and EndString are eligible 5 letter English words
 - Call WordLadderSolver with StartString and EndString as arguments
 - Output the Word Ladder in the required format
- Exit

Main Initializes a dictionary (Arraylist of strings and checks) Dictionary () Initializes the vector array for the word ladder word Ladder Solver() User imputs starting word and target word * must be words in dictionary Inputs Uses a DFS algorithm to generale the ladder between the two input words MakeLadder Informs the user that a ladder doesn't exist FAILURE No SwenLadder Exception SUCCESS Prints the word ladder out to the console Print solution() Functional Block Diagram

Design Rationale

The design approach is to make a WordLadderSolver have a Dictionary, a Dictionary made of Words. An alternative was just to make the dictionary a group of strings, however, by making a word class, we were able to give the words attributes such as visited that would make it simpler when trying not to make an infinite loop or debugging in a programming perspective (there really would not be a change in the user perspective as long as it worked). In terms of flexibility, our WordLadderSolver (with the change of some variables) can easily be modified to work with a change of word length (as opposed to five) as long as the dictionary has more than 5 letter words. In the end, we believe that our project adheres to the principles of good design in the fact that it implements an OOD with a solid foundation as well as hides the implementation to the client programmer to the point where if we decide to change our data structure (hashtable to linked list, etc.) the program will still function properly without any changes from the client programmer's perspective.