Rationale

A physical dictionary would be a real world representation of our dictionary class. Like a real dictionary, our class contains an ordered list (by alphabetical order) of valid words that we can reference to see if words from an external party are real. When thinking of alternatives ways to approach this problem, we contemplated on using an iterative approach rather than a recursive approach for the MakeLadder( ) method. We would have used this approach had the lab manual not told us we had to use recursion for this method. Iteration makes testing easier than recursion since recursion maintains some information hidden to the program during execution. In addition, code is easier to follow for an iterative method. The advantage of the recursive method is that it makes the method more elegant since it does not require as many nested loops as an iterative method. Our design offers flexibility to the length of a word. Throughout our program, we don't assume that each word is 5 letters long, except in the dictionary class. With a slight modification to the extractWords ( ) method, we can handle dictionaries of different word sizes. Our program has a low level of coupling and higher level of modularity which makes our program easier to test. The different modules handle different aspects of a program. We have high cohesion since there is a lot of dependence between elements of the same module which also makes testing easier.