Word Ladders

When I was a sophomore in college, I found myself in what would turn out to be the most influential course that I've ever taken. Everything about the course was in some way exceptional, but the professor particularly so. She made what could have been an ordinary English Composition class into a critical examination of thinking. One of the recurring themes in the class was *making connections* - the process that allows us to associate things with one another and to see relationships among different things. So, in honor and memory of O.A.L., this assignment is all about *making connections*.

Problem Overview

The focus of the assignment is to implement a word connection game that has been played in one variation or another for almost 150 years. The object of the game is to transform a *start word* into an *end word* of the same length by a sequence of steps, each of which consists of a one-letter change to the current word that results in another legal word. Charles Lutwidge Dodsgon (Lewis Carroll) invented this game and called it "Doublets." It's now more commonly known as Word Ladders.

Consider the following examples.

```
cat, can, con, cog, dog
cat, bat, eat, fat, gat, hat
clash, flash, flask, flack, flock, clock, crock, crook, croon, crown, clown
```

Each is a valid word ladder from the start word to the end word since the start and end words are the same length and each word in between is exactly one letter different from the previous word.

The game is usually played so that each player tries to find the *shortest* word ladder between two words. The shortest ladder would, of course, depend on the *lexicon*, or list of words, being used for the game. Using the SOWPODS word list (see below), word ladders with minimum length for the start-end pairs above would be:

```
cat, cot, dot, dog
cat, hat
clash, class, claws, clows, clown
```

Implementation Details

You must implement your solution to the assignment in terms of `WordLadderGame`, an interface that specifies all the behavior needed to calculate word ladders, and `Doublets`, the shell of a class that implements the `WordLadderGame` interface. You must provide a correct implementation of the `Doublets` class by completing its constructor and providing a correct implementation of each method. You must not change the `WordLadderGame` interface in any way. You must meet all the requirements specified and implied by the Javadoc comments in these files. You may add as many methods as you would like, and you may add as many nested classes as you would like. Although you may import other classes that are part of the JDK, the imports already provided are the suggested ones that you will need.

Downloads

You can download the necessary starter code and resources here:

- WordLadderGame.java
- Doublets.java
- ExampleClient.java
- WordLists.jar

The `ExampleClient` class illustrates basic calls to the `WordLadderGame` methods, and it also demonstrates how to associate a text file contained in WordList.jar with an `InputStream` object. Text files containing different word lists of various sizes are provided in the Java JAR file WordLists.jar. JAR files can be opened by most common file compression and archiving utilities, as well as jGRASP via the Project menu. You can also extract the contents of a JAR file on the command line by issuing the following command:

```
jar xf WordLists.jar
```

Acknowledgements

Word search games of various sorts are popular CS 2 assignments because they bring together several important topics all in one place. This version of the word search problem owes thanks to Owen Astrachan and others.