Lab Assignment 2
When Logic and Proportion Have Fallen Sloppy Dead

Lab Objectives

- Understand and apply fundamental principles of object-oriented design and programming.
- Understand, design, and apply appropriate collections for solving a problem.
- Be challenged to think and make connections.

Lab References

- OOD, OOP
- Collections, Comparison, Searching, Strings, Optimization

Problem Description

When I was a sophomore in college, I found myself in what would turn out to be the single 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 an examination of thinking. One of the recurring themes in the class was "making connections" – the process of thought and intellect that allows us to connect things together and see relationships among different things. So, in honor and memory of O.A.L., this assignment is all about making connections.

The focus of the assignment is to implement a word connection game that has been played in one variation or another for at least 130 years. The game is to transform a start word into an end word of the same length by a sequence of one-letter changes at a time, where each change produces a legal word. We'll call this sequence of word transformations a *Dodgson sequence*.

For example, given the starting word "clash" and the final word "clown" a possible Dodgson sequence would be:

clash, flash, flask, flack, flock, clock, crock, crook, croon, crown, clown

A fun variation on the game (and one that extends the whole idea to another level) is to choose start and end word pairs with some inherent connection to each other, like being synonyms, antonyms, or having some other relationship. Here are some example Dodgson sequences that illustrate the idea:

head, heal, teal, tell, tall, tail
door, boor, book, look, lock
bank, bonk, book, look, loon, loan

Okay, now think about that for a second. If you also required that each *step* in the sequence be meaningfully related to its predecessor, then you would have the basis for addressing some interesting problems. For example, you could adapt that basic strategy to the computation of Erdös numbers and Bacon numbers. You could also have a core strategy for "connectedness" applications on social networks such as Facebook, LinkedIn, and Twitter. The intent of these applications could range from suggesting possible "friends" to data mining for a law enforcement or intelligence agency. (Ever wonder what your Facebook degree of separation is from a person of interest in an ongoing sleeper cell investigation? You can bet the NSA knows.). The point here is that important connections are everywhere; a word game from the 1800s is related to important, applied problems today.

Now, back to 2210 …

What you must do:

You must write a Java program that generates the **shortest** Dodgson sequence for given start and end word combinations. Your program must take its input from a text file named diller.txt, which contains an unspecified number of start and end word pairs, one pair per line. Your program must output the *shortest* Dodgson sequence for each line of input. To qualify as a word, a character string must appear in some agreed-upon lexicon. For this assignment, you must use the provided subset of the SOWPODS lexicon, the official lexicon for international Scrabble® tournaments.

Your solution **must** involve the following files, at a minimum.

1. **Assign2.java** – This class drives the execution of your solution. Executing the main method of this class must generate the shortest Dodgson sequence for every word pair listed in diller.txt. The output must be a comma-separated list of strings that begins with the start word and ends with the end word (see the examples above), one sequence per line. NOTE: You must observe this output format since automatic string matching will be used to score the output of your code.

2. **Dodgson.java** – This class is the Dodgson sequence "generator." It must contain all the logic needed to transform one string into another string (start word, end word) while obeying the Dodgson sequence rules: (1) The start word, end word, and all intermediate words must be of the same length (contain the same number of characters). (2) A valid transformation is one in which a given word is transformed into a new word by changing only one letter. (3) To qualify as a "word" a string must be in the agreed-upon lexicon (either sowpods.txt or sowpods_small.txt).

3. **ILexicon.java** – This interface describes all the services that can be used to access the lexicon. This interface is provided to you and MUST NOT BE CHANGED.

4. **Lexicon.java** – This class must implement ILexicon and will provide access to the agreed-upon lexicon. You are free to add any method you like beyond the ILexicon methods.

5. **sowpods_small.txt** – This file contains the words that compose the lexicon to be used for qualifying strings as words.

6. **diller.txt** – This file is the required input file of start- and end-word pairs (one per line, separated by blanks).

You may implement and use other interfaces and classes at your discretion. Your program should run with the full lexicon from sowpods.txt. But, if the 3MB size causes you too many problems, only 5 points will be deducted for having to use sowpods_small.txt.

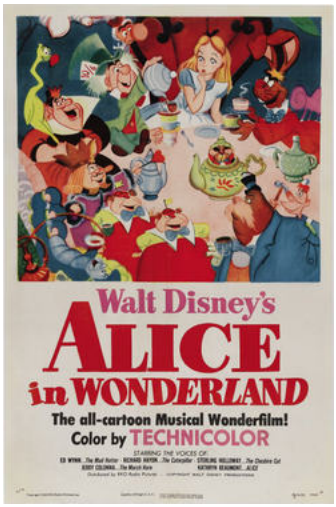
Lab Turn-In

There are multiple submissions associated with this assignment. Refer to the first page of this document and Canvas for all deliverables and dates. For each deliverable, however, a new requirement beginning with this assignment is that you include your user id as the first part of any files that you upload to Canvas. For example, I might name one of my submissions hendrtdAssign2.zip.

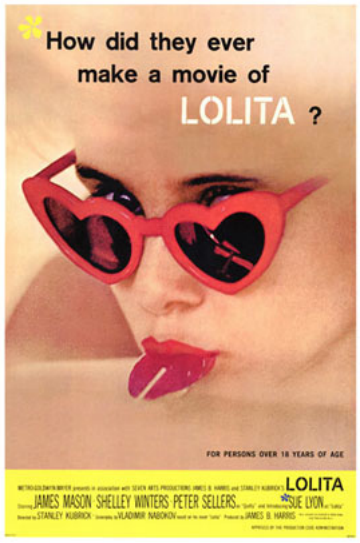
Extra Credit

If you really want to have some fun and challenge your grey cells at the same time, you can opt to implement a second game mode in which you take a single word as input (the start word) and output a Dodgson sequence that transforms it into some *related* word that your program chooses. The relation would have to be something like synonym, antonym, or semantic association (bank – loan). To be considered for the extra points you must correctly implement the required part of the assignment. You can earn up to 10 points for this game option.

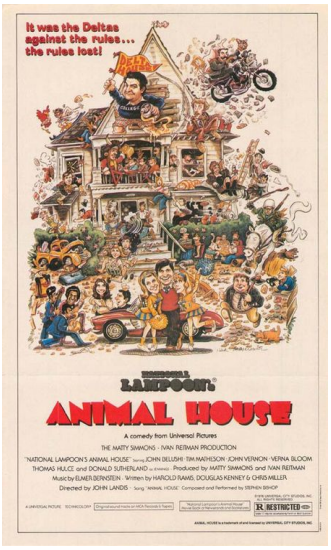
And, just for fun and maybe as a team icebreaker, the assignment carries an additional, first-come, first-served 10-point bonus in honor of my old English professor. The first team in the class to send me an email that correctly connects the content of this assignment (Dodgson sequences) to the game "Six Degrees of Kevin Bacon" will earn the points. Only connections that adhere to the following rules will be eligible for the points. (1) Intermediate steps in the connection **must** include the following three films/books: *Alice in Wonderland*, *Lolita*, and *Animal House*. The connection **must** be specified in the spirit of a "Dodgson sequence" – that is, a linear connection just like you have to connect words. This connection is not lexical. It is entirely conceptual, and you must be able to write a narrative description of the linear sequence of connections something Dodgson sequence >> Alice in Wonderland >> Lolita >> Animal House >> Six Degrees of Kevin Bacon. There may be other intermediate steps than those shown. (And just to go ahead an answer the inevitable question: There is no illegal/immoral sexual activity involved in the connections.)



Alice in Wonderland



Lolita



Animal House