**IndexMaker with a Map**

Original lab from: Litvin, Gary and Litvin, Maria. *Java Methods: Object Oriented Programming and Data Structures.* Skylight Publishing, 2011. pp 370-374.

In this lab you will write a program that reads a text file and generates an index of line numbers for each word. All the words that occur in the text should be listed in the index in upper case in alphabetical order. Each word should be followed by a list of all the line numbers of lines that contain that word. Study the input file fish.txt on the left and how it relates to the output file fishIndex.txt on the right. (Note the commas!)

A 12, 14, 15

ARE 16

BLACK 6

BLUE 4, 7

CAR 14

FISH 1, 2, 3, 4, 6, 7, 8, 9, 16

HAS 11, 14

LITTLE 12, 14

LOT 15

NEW 9

OF 16

OLD 8

ONE 1, 11, 14

RED 3

SAY 15

STAR 12

THERE 16

THIS 11, 14

TWO 2

WHAT 15

One fish fish

two fish fish

red fish

blue fish.

Black fish

blue fish

old fish

new fish.

This one has

a little a star.

This one has a little car.

Say! What a lot

of fish there are.

Before, the data structure was an ArrayList of IndexEntries. This time, the DocumentIndex *isa* map. Would you choose a HashMap or a TreeMap?

In Java, objects must map to other objects. However, the objects may be another data structure. This time we chose to map to Set. Why choose a Set, not an ArrayList? Would you choose a HashSet or a TreeSet?

Study this UML:

TreeMap<String, TreeSet<Integer>>

*isa*

IndexMaker

DocumentIndex

You will be able to copy parts of your old IndexMaker lab (in the "05 Collections" folder) into IndexMakerMap.

When you are finished coding, be sure to set a breakpoint and look at your tree of mappings, which map Strings to sets of Integers.

Part of the DocumentIndex class is on the back of this page.

class DocumentIndex extends TreeMap<String, TreeSet<Integer>>  
{  
 /\*\* extracts all the words from str, skipping punctuation and whitespace   
 and for each word calls addWord(word, num). A good way to skip punctuation   
 and whitespace is to use String's split method, e.g., split("[., \"!?]") \*/  
 public void addAllWords(String str, int lineNum)   
 {

}  
  
 /\*\* Makes the word uppercase. If the word is already in the map, updates the lineNum. Otherwise, adds word and ArrayList to the map, and updates the lineNum \*/  
 public void addWord(String word, int lineNum)  
 {

}  
   
 public String toString()   
 {

}  
}

Remember foundOrInserted ? Why don’t you have to implement foundOrInserted?