**Authors and Novels with a Map**

The input file lists an author followed by one novel:

Flaubert, Madame Bovary  
 Dostoevski, Crime and Punishment  
 Tolstoy, Anna Karenina  
 Stendhal, The Red and the Black  
 Tolstoy, War and Peace  
 Dostoevski, The Possessed  
 Dostoevski, The Brothers Karamazov  
 Flaubert, A Simple Heart  
 Dostoevski, The Grand Inquisitor  
 Flaubert, Memoirs of a Madman  
 Flaubert, Sentimental Education  
 Tolstoy, The Death of Ivan Illyich  
 Tolstoy, The Kreutzer Sonata

The output file lists an author followed by all of his/her novels in alphabetical order. The entire list is alphabetized by author:

DOSTOEVSKI: Crime and Punishment, The Brothers Karamazov, The Grand Inquisitor, The Possessed  
FLAUBERT: A Simple Heart, Madame Bovary, Memoirs of a Madman, Sentimental Education  
STENDHAL: The Red and the Black  
TOLSTOY: Anna Karenina, The Death of Ivan Illyich, The Kreutzer Sonata, War and Peace

In the previous AuthorsNovels, the data structure was an ArrayList of Author objects, each of which had two fields, and one of the fields was an arrayList. Each list had to be alphabetized. Complicated!

Good news! Maps greatly simplify the data structure. This time, we will make an AuthorsMap that will map Authors to a Set of books. Find the Map and the Set in the output file above.

Study this UML:

TreeMap<String, Set<String>>

*isa*

AuthorsNovelsMap

AuthorsMap

Why did we choose a TreeMap? What kind of Set should you instantiate in order to store the novels?

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

The previous AuthorsNovels is in 05Collections. You may be able to re-use part of the code.

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

The AuthorsNovelsMap class is the driver class and is already written. Its main method prompts the user for the names of the input files, opens the input file, creates an output file, reads and processes each line from the input file, then writes the resulting AuthorsMap in the output file. You don't have to code for reading or writing files in this lab.

**AuthorsMap**. The AuthorsMap represents all the authors with all their novels. Note the extends TreeMap<String, Set<String>> in the header. It means that you get the TreeMap and all its methods for free. Use it! Don’t make a new field for it! To access the TreeMap, you can use the this keyword. You don’t need explicit constructors, but since it is good practice to put in constructor(s), you will write one default constructor. It uses the super keyword.

class AuthorsMap extends TreeMap<String, Set<String>>  
{  
 /\*\* when you extend a class, the constructor is optional \*\*/  
 public AuthorsMap()  
 {  
 super();  
 }  
   
 /\*\* extracts the author and book from oneLine.  
 calls addAuthorOrNovel   
 \*/  
 public void readOneLine(String oneLine)   
 {   
  
 }  
   
 /\*\* either inserts a new Author mapping, or updates a previous Author mapping  
 \*/  
 public void addAuthorOrNovel(String name, String book)  
 {  
  
 }

public String toString()  
 {  
  
 }  
}