**Authors and Novels**

The input file lists an author followed by one novel:

Flaubert, Madame Bovary  
Dostoevski, Crime and Punishment  
Tolstoy, Anna Karenina

Dostoevski, The Possessed

Dostoevski, The Possessed

The output file lists an author followed by all of his/her novels. The list is alphabetized by author. The novels are not alphabetized; novels are added in the order that they are processed. The list of novels should not contain duplicates.

DOSTOEVSKI: Crime and Punishment, The Possessed  
FLAUBERT: Madame Bovary  
TOLSTOY: Anna Karenina

What resource class(es) do you have to write?

What are the advantages and disadvantages of arrays vs. ArrayLists?

How would you process the input file to result in the output file?

You are going to write two user-defined classes: AuthorList and AuthorEntry. The application uses ArrayLists in two ways: AuthorList *isa* ArrayList of AuthorEntrys and each AuthorEntry *hasa* ArrayList field that holds the books. Study this UML:

ArrayList<AuthorEntry>

*Comparable<AuthorEntry>*

*isa*

String

*isa*

*hasa*

AuthorEntry

AuthorsNovels   
 (the driver)

AuthorList

*uses*

*uses*

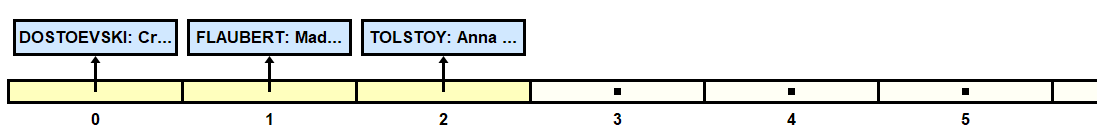
*hasa*

ArrayList<String>

The AuthorsNovels 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 saves the resulting AuthorList in the output file. You don't have to code for reading or writing files in this lab.

**The algorithm.**

Using a listIterator, insert or update an AuthorEntry object in the ArrayList. The AuthorEntry objects are always stored in alphabetical order.



What methods will you use?

List all the cases:

**AuthorList**. The AuthorList represents the list of authors with all their novels. Note the extends ArrayList<AuthorEntry> in the header. It means that you get the ArrayList and all its methods for free. Use it! Don’t make a new field for it! To access the ArrayList that you get from the hierarchy, you can use the this keyword. The explicit constructor is optional, but since it is good practice to put in constructor(s), we write one default constructor. It uses the super keyword.

class AuthorList extends ArrayList<AuthorEntry>  
{

/\*\* you get an ArrayList for free \*\*/

public AuthorList()  
 { super(); }

**AuthorEntry**. An AuthorEntry object represents one author and his/her books. Note the implements in the header! What must you do to implement Comparable?

class AuthorEntry implements Comparable<AuthorEntry>  
 { //fields  
 private String name;  
 private ArrayList<String> novels;  
   
 //two constructors. argument n may be in lowercase.   
 public AuthorEntry(String n)  
 { }   
 public AuthorEntry(String n, String book)  
 { }

**Text Files.** Two files, input.txt and input2.txt, have been given to you.

**Extension**: Rewrite AuthorEntry with this header:

class AuthorEntryExt extends ArrayList<String> implements Comparable<AuthorEntryExt>

Now the novels are not stored in an ArrayList field, but in the ArrayList that you get from the hierarchy. Use it.