1.) Using the attached catalog.txt file, create a program that begins by loading the file onto an arraylist of Album classes called Catalog. Each Album class consists of 3 attributes: artist name, album name, and an arraylist of Track objects, each containing the name of a song on the album.

2.) Implement the Comparable interface for the Album and the Track classes, defining their own compareTo() methods.

3.) Sort the Album arraylist using Collections.sort(), and display the sorted Albums by album name. For each Album, display the album name, artist name, and display all the Tracks (i.e. songs) in alphabetical order (use the sort() method too)

4.) Add another sort of the Album arraylist, by **artist name**. To implement this sort, you will need to implement a Comparator interface, and create a Comparator class. Save the results of this sort in a separate ArrayList from the one sorted by album name. **Print a dividing line of asterisks, and then display all the Albums by artist name. For each Album, also display the album name, and all its tracks (songs) in alphabetical order. For more hints, see pg. 661 in CH. 14 of Big Java, 5th Edition.**

5.) Display a menu for the user to select from the following options, once the Album arraylist is fully populated:

1. Search by Album Title
2. Search by Artist
3. Add album to catalog
4. Quit

5.) For the Album search, first sort the ArrayList of Albums by album name, using the Comparable interface and compareTo methods in the Album class. Then, use the binary search method for Collections, Collections.binarySearch(), to find the album requested by the user.

6.) For the artist search, sort the ArrayList of Albums by artist name, using the Comparator interface and your comparator class. Once sorted by artist, use the binary search method for Collections, Collections.binarySearch() to find all the albums for a particular artist requested by the user. Display the names of all the albums that have been created by the artist.
***Hint: Since there are multiple albums by one artist, the value returned by the binarySearch() method is arbitrary in the list of the same artist. You will have to keep getting the rest of the albums by searching above and below that arbitrary index, until the artist is different. Use the ArrayList.get()method (see page 349 of Big Java, 5th Edition, for an exmple of a loop to sequentially go through an arrayList.)

7.) Implement the Scanner and File classes to read the Album file.

8.) Implement the FileWriter and PrintWriter classes to write to the Album file.

9.) Catch exceptions such as IOException and FileNotFound. Thus, prompt the user for the name of the input/output file. Give the user a message that says "Enter catalog2.txt for the file input."

10.) If the user selects the "add album to catalog" option, then prompt the user for all the information that makes up an Album object, and once created, add the new Album to the ArrayList of Albums. Don't forget to re-sort the ArrayList of Albums. Don't forget to permanently add the album to the external file, using the FileWriter and PrintWriter classes. It may be easier to create a .toString() method in the Album class, and then use it to write the new Album record to the file. (Ex.: outputStreamName.println(objectName) will invoke the .toString()) behind the scenes.)