CSCD 211
Lab 2

PROGRAM SPECIFICATIONS

In this lab you will simulate a bookshelf full of books. The books on your bookshelf will be contained in an array. You will be able to remove books from the shelf, add books to the shelf, organize the books on the shelf, find a book on the shelf, and various other functions. See menu below for more detail. The initial bookshelf will come from a file with the name specified by user. The initial file format is explained below:

A bookshelf is comprised of an array of type Book (Base class). A Book is comprised of a title, number of pages, an edition number, an ISBN and an array of type Author. Information for each class is specified below.

CLASS SPECIFICATIONS

Author

An **Author** class contains the following attributes as **private** instance fields:

- Last Name - String
- First Name - String
- Year - int

The class should have the following instance behaviors:

- A default constructor(initializes last name to Nerd, first name to Ima, and year to 2014)
- An explicit value constructor (passed the first name, last name, and year)
- A toString (returns a String formatted as follows:**Author: Stu Steiner, Year: 2014**)
- A compareTo method that compares **by last name** then first name then year
- Accessor (get) methods for each field *as you deem necessary*
- Modifier (set) methods for each field *as you deem necessary*
- Helper methods *as you deem necessary*

Book

A **Book** class that contains the following attributes as **private** instance fields:

- Title - String
- Number of pages – int
- Edition - String
- ISBN – String
- An array of type Author

The class should have the following instance behaviors:

- A **private** disabled default constructor – empty – you will never have a book
- An explicit value constructor that is passed
    - The title of the book
    - The ISBN
    - The number of pages in the book
    - The edition information
    - The author information – Note: You will have to develop a way to handle the Author information.
- A toString method that returns a String that contains the Book information in the following format:
  Title: Building Java Programs: A Back to Basics Approach
  Edition: 1st
  ISBN: 978-0321382832
  Author: Stuart Reges, Year 2008
  Author: Marty Stepp, Year 2008
  Pages: 896

- A compareTo method that is passed a Book object (Must use the parameterized type <Book>) compares **by book title** and then by edition. The method should return 0 if the two Book objects are the same, a negative integer value if the current Book is less than the Book passed in, and a positive integer value if the current Book is greater.

- An equals method checks to see if the book title and the ISBN are equal
- Accessor (get) methods for each field *as you deem necessary*
- Modifier (set) methods for each field *as you deem necessary*
- Helper methods *as you deem necessary*

TextBook

- Derived from Book
- Contains a String for Subject
- Contains a EVC that receives the values for a book and the subject
- Any methods you deem necessary
- toString adds the type then calls Book toString and appends the following: Subject: Subject Info

Novel

- Derived from Book
- Contains a String for type (fiction/non-fiction)
- Contains a EVC that receives the values for a book and the type
- Any methods you deem necessary
- toString that adds Novel: type and then calls Book toString

CSCD211Lab2

- A count method that is passed the File object for the input file.
  - Creates a Scanner object
  - Reads through the file and counts the number of books – this will be done by examining the type as novel or textbook
  - Close the file when finished

- A readBooks method that is passed the File object the array of type Book. The method should:
  - Opens the file specified
  - Read the Book data from the file and create Book objects
  - Assign each Book object to an array of type Book
  - Close the file when finished

- A printBooks method that is passed a PrintStream and an array of type Book.
  - The PrintStream object will either be System.out or a reference to a file.
  - Write the count of Books followed by each Book (make sure you use the toString method provided by the Book class for this)

- A menu method that is passed a Scanner object
  - Menu choices are below
  - You must ensure values entered are within range

  MENU SPECIFICATIONS

  1. Print all Books to the screen
  2. Print all Books to the User Specified file
  3. Sort the books by book title then by edition (Hint: compareTo)
  4. Sort the books by Subject, ISBN, and title (Comparator)
  5. Sort the books by Author (Comparator)
  6. Find and print, to the screen, a book in the array (equals method)
  7. Find and print, to the screen, all books of the same type (Novel or Textbook)
  8. Quit

- An executeChoice method that is passed the array of type Book, the choice, and a Scanner object. This will execute the appropriate set of methods to complete the required tasks.

- The main method – see the Java file. You can't change my main method – If unsure just ask

INPUT FIlE SPECIFICATIONS

The input file will have at least one entry. Each entry will be formatted as follows:

```
Type meaning novel or textbook all lowercase

Appropriate information depending on the book type from above

Book title
Edition
ISBN
Number of pages
Number of authors
Author first name<space>Author last name<space>Year
```

Here is a small sample input file to help clarify:

textbook
programming
Java Software Solutions
6th Edition
978-0321532053
806
2
John Lewis 2006
William Loftus 2008
textbook
programming
Ruby on Rails
1st Edition
978-1934356166
167
1
Bruce Tate 2007
novel
non-fiction
Dad is Fat
1st Edition
978-0385349055
288
1
Jim Gaffigan 2013

Be sure your input file contains data so your file can be properly tested – especially for the different sorts.

Output file – name specified by the user

```
Book count: 3

Information on Book 1
```

Subject: Programming
Title: Java Software Solutions
Edition: 6th
ISBN: 978-0321532053
Author: John Lewis, Year 2006
Author: William Loftus, Year 2008
Pages: 806

```
Etc
```

TO TURN IN

A **zip** file that contains:

- All java source files needed to compile your code
- All input and output files

You zip will be named your last name first letter of your first name lab2.zip
(Example: steinerslab2.zip)

Get started ASAP!