CS 152 Computer Programming Fundamentals
Project 7: Author, Book, and Library Objects

Brooke Chenoweth

Fall 2026

To give you some practice creating and using classes, you are going to write three classes for this project: `Author`, `Book`, and `Library`. You will have to create your classes from scratch, but don't worry, I give a detailed description of what I expect below. The `Library` class will build on your `Author` and `Book` classes, so don't even start it until you are done implementing them.[1]

I am giving you testing code in `AuthorBookTester.java` and `LibraryTester.java`[2] and a class containing some constants in the `AuthorBookConstants.java` file. Do not change these files. Please note: these files will not compile without your classes.

1 Author class

An `Author` is a writer of books. An author has a given name, a surname[3], and birth/death years.

1.1 Constructors

The `Author` class has one constructor:

- `Author(String surname, String givenName)` – Creates a new Author with a name, but birth and death years not set. Please note that the *first* argument is the Author's *surname*.

1.2 Methods

An `Author` has at least the following methods: (You can make more if you feel you need to, for helping your other methods, but I will be calling and testing these.)

- `getBirthYear` – Takes no arguments and returns an int representing the author's year of birth.

[1] In fact, you won't even be given the Library test code until you've submitted your initial Author and Book implementation, so start early.

[2] You won't get this file until after you submit Author and Book for part 1.

[3] Or family name. In America, this usually is the last name.

- `getDeathYear` – Takes no arguments and returns an int representing the author's year of death.

- `setLifespan` (birth year version) – This is an overloaded method. Takes int for birth year and sets the author's year of birth. It must be a valid year. (See section 4.1 for definition of valid year.) Returns nothing.

- `setLifespan` (birth and death years version) – This is an overloaded method. Takes two ints for birth year and death year and sets both the author's year of birth and death. Both must be valid years, and the year of death must be after the year of birth. (See section 4.1 for definition of valid year.) Returns nothing.

- `hasSameName` – Takes another Author as an argument and if two authors have the same given names and surnames it should return true, otherwise it should return false.

  Bonus: Sometimes authors do not always write out their entire first name. For example Julian Barnes and J Barnes could be considered equivalent. However, Julian Barnes and John Barnes should not be. For bonus points, make your hasSameName more flexible by returning true if a first initial and surname match.[4]

- `toString` – Takes no arguments and returns a string representation of the author, of the form: `Barnes, Julian`

  This is a string of the surname, a comma and a space, and the given name.

- `getLongString` – Takes no arguments and returns a fancier string representation of the author, including birth and death years if they have been set.

  If neither birth nor death years are set it should return:
  `Heaney, Seamus`
  If the year of birth (but not death) is set it should return:
  `Heaney, Seamus (born 1939)`
  If both the years of birth and death have been set, it should return:
  `Heaney, Seamus (1939-2013)`

2 Book class

A `Book` is our main object. Books in our system have a date (year) of publication, a title, an ISBN, and a single Author (`Author` object) who wrote them.[5]

[4]This probably isn't what you would want in general, since J Doe matching both John Doe and Jane Doe could have unexpected behaviour in a larger library system. A real system probably would have separate methods to compare with exact match vs initial match, but this is just a little challenge for this assignment.

[5]I realize that actual books may have more than one author, but we're simplifying for the purposes of this assignment.

2.1 Constructors

`Book` has three constructors:

- `Book()` – The default (no parameters) constructor, creates a new Book with no title, year of publication, Author, or ISBN set.
- `Book(String title)` – Creates a new Book with a title, but no year of publication, Author, or ISBN set.
- `Book(String title, Author author)` – Creates a new Book with a title and an Author (as an `Author` object) but no year of publication or ISBN set.

2.2 Methods

A `Book` has at least the following methods: (Again, you can make more if you feel you need to, for helping your other methods, but I will be calling and testing these.)

- `setTitle` – Takes a String argument and sets the book's title. Title must not be empty. Returns nothing.
- `getTitle` – Takes no arguments and returns a String representing the book's title.
- `setAuthor` – Takes an Author argument and sets the book's author. Returns nothing.
- `getAuthor` – Takes no arguments and returns the Author object for the author who wrote the book.
- `setPublicationYear` – Takes an int argument and sets the year the book was written/published. It must be a valid year. (See section 4.1 for definition of valid year.) Returns nothing.
- `getPublicationYear` – Takes no arguments and returns the year the book was written.
- `setISBN` – Takes a String argument and sets the ISBN for the book. This must be of length 13 or length 10. (We will be omitting dashes/spaces in our tests, so you can just check the length of the String.) Returns nothing.
- `getISBN` – Takes no arguments and returns a String representing the ISBN of the book.
- `hasSameAuthor` – Takes a Book argument and returns true if this book has the same `Author` as the book that was passed in as an argument, otherwise return false. We'll consider two authors to match if they have they have the same name. (Hint: You already wrote a method to check for this.)
- `isSame` – Takes a Book argument and returns true if this book is the same as the other book. We will say that two `Book` objects are the same if their ISBNs are equal.

- `toString` – Takes no arguments and returns a String representation of this Book.

  If only a title has been set, return a string of the form:
  `The Arcades Project.`
  If a title and author are set:
  `The Arcades Project.  Benjamin, Walter.`
  If a title and author and year have been set:
  `The Arcades Project (2002). Benjamin, Walter.`

  Note the punctuation in the examples above, it must be exact to pass the tests.

3 Library Class

A Library object keeps holds multiple Book objects, tracks how many copies of each are in the libary, and how many are currently checked out.

3.1 Member Variables

- An integer holds the number of unique books in the library.
- An array of Book objects holds the unique books in the library.
- An array of integers holds how many copies of each book are in the library.
- An array of integers holds how many copies are currently on the the shelf (checked in) for each book.[6]

3.2 Constructor

The `Library` class has one constructor:

- `Library(int capacity)` – Initialize an empty library with the ability to hold as many unique books as specified by the capacity argument.

3.3 Methods

- `getTotalCopies` – Takes no arguments and returns the number of total copies of all books that exist in the library.
- `getTotalCheckedOut` – Takes no arguments and returns the number of total checked out book copies at the current time.
- `getStatus` – Takes no arguments and returns a string representing the status of the library, in the format:

[6] We don't need to separately track how many books are checked out since we can compute that value from total copies and checked in.

```
Total unique books: 26
Total number of copies: 40
Total checked out: 0
```

- `addBook` – Takes a single book argument and adds it to the library. If the book is already present, it should add another copy of the pre-existing book. If the book is new, it should be added to the end of the books. The newly added book will be available on the shelf, that is, checked in. Returns nothing.
- `addBooks` – Takes an array of Book objects and adds them all into the library. Note: there may be other books in the library already so you cannot simply overwrite the entire books array, but rather must add these at the end of the current library. Make sure to include one copy of each book. Returns nothing.
- `checkOut` – Takes a book argument and checks it out from the library if the book exists in the library and there are remaining copies which haven't been checked out. Return a string denoting success or failure, one of the following:

  ```
  Book not known in library.
  ```

  ```
  All of our copies are already checked out.
  ```

  ```
  Checked out!
  ```

- `checkIn` – Takes a book argument and checks it in to the library if the book exists in the library and there are copies which have already been checked out. Return a string denoting success or failure, one of the following:

  ```
  Book not known in library.
  ```

  ```
  All of our copies are already checked in.
  ```

  ```
  Checked in!
  ```

- `toString` – Takes no arguments and returns a string representing the entire library collection, including status, where book numbering starts at 0, and the numbers after the colon denote currently available copies and total copies:

```
0. Ulysses. Joyce, James. : 1/1
1. The Great Gatsby. Fitzgerald, F. Scott. : 1/2
2. Lolita. Nobokov, Vladimir. : 1/1
3. Brave New World. Huxley, Aldous. : 1/1
4. The Sound and the Fury. Faulkner, William. : 2/2
5. Catch-22. Heller, Joseph. : 1/1
6. Darkness at Noon. Koestler, Arthur. : 1/1

Total unique books: 7
Total number of copies: 9
Total checked out: 1
```

- `numBooksByAuthor` – Takes an Author object and returns how many unique books exist in the library for that author.
- `listBooksByAuthor` – Takes an Author object and returns a String that lists the unique books which exist for that author, in standard book format:

  ```
  Pitch Dark. Adler, Renata.
  Speedboat. Adler, Renata.
  ```

  Include a line break `'\n'` after each title (including the last one). If no books are found this method should return:

  ```
  No books by Adler, Renata.
  ```

- `listBooksByTitle` – Takes a String argument and returns a String that lists the unique books which contain the given string within their titles, without regard for case, in standard book format:

  For example, a search for `"of"` might return:

  ```
  A Portrait of the Artist as a Young Man. Joyce, James.
  The Grapes of Wrath. Steinbeck, John.
  The Way of All Flesh. Butler, Samuel.
  The Wings of the Dove. James, Henry.
  ```

  Include a line break `'\n'` after each title (including the last one).

  If no books were found when searching for `"cloud atlas"` this method should return:

  ```
  No books with "cloud atlas" in the title.
  ```

- `deleteBook` – Bonus problem! This method deletes all copies of a book entirely from the library, if it exists. It must update all arrays to not leave a hole in the sequence. Return a string denoting success or failure, one of the following:

  ```
  Book removed.
  ```

  ```
  Book not known in library.
  ```

  If you do not want to add the bonus functionality, just write a `deleteBook` method that takes a Book argument and returns a String so that the tests can compile.

4 General Notes

4.1 Valid Years

A valid year is one after the year -2000 and before the year 2029. (Some books do announce their publication in advance, though rarely more than a few years out. Most people have not forecast their births or deaths more than a few years out). Also, there is no year zero, so that would also be an invalid year.

4.2 Invalid Values

When trying to run a setter method with an unallowable variable, you should leave the previous value unchanged. Printing error messages when this happens is a good idea, however the text of them is up to you.

4.3 Uninitialized Values

There are a series of constants provided for you in the `AuthorBookConstants` class. You should use these variables for properties of books and authors that have not yet been set.

4.4 Access Modifiers

The member variables for your classes should be private. The methods and constructors listed above should be public. If you make any additional methods, it is up to you which access modifier you use, but do pick one. Consider if anyone outside of your class might need access to your helper method.

5 Testing Your Code

5.1 Writing Your Own Tests

The provided test code will not compile until you have implemented the `Author` and `Book` classes. I strongly suggest that you create your own main method in each of these two classes to test your methods as you are developing them. Java will allow you to put a main method into any class you like, so have at it!

5.2 Provided Test Code

The graders will use the `LibraryTester` class[7] to test your code.[8]

The expected output if you pass all the tests looks like the following. (There may be additional output if you printed some error messages for invalid values, but it will be something like this.)

```
*** TESTING AUTHOR/BOOK IMPLEMENTATIONS ***
Attempting constructors:
  - Author constructors seem functional:  2/2
  - Book constructors seem functional:    2/2

Attempting simple getters/setters:
  - getTitle/setTitle:                  2/2
  - getAuthor/setAuthor:                2/2
  - get/setPublicationYear:             2/2
```

[7] Provided after you submit part 1 on Canvas

[8] They will use `AuthorBookTester` for testing your part 1 submission.

```
  - get/setISBN:                          2/2
  - getBirth/getDeath/setLifespan:      4/4

Attempting un-set values register correctly:
  - un-set values:  5/5

Attempting bad-value setters:
  - invalid publication year:             3/3
  - empty title rejected:                 2/2
  - ISBN of incorrect length rejected:    2/2
  - year of birth in super past rejected: 2/2
  - year of death before birth rejected:  2/2

Comparisons:
  - author comparison tests:          4/4
  - *bonus* author comparison tests:  5/5
  - sameAuthor (across two books):    2/2
  - comparing two books:              2/2

String representations of authors and books:
  - string for an author's name:         2/2
  - string for an author's name + info:  3/3
  - string for a book's name + info:     5/5

Overall Score: 55/50
   (can be up to 55 with bonus)

*** TESTING LIBRARY IMPLEMENTATION ***
Let's calculate some facts about the library...
 - getTotalCopies works for empty library: 1/1
 - getTotalCheckedOut works for empty library: 1/1
 - getStatus  works for empty library: 1/1

Let's add a set of books, and add more books...
 - addBooks seems to work: 1/1
 - addBook seems to work for duplicates: 1/1
 - addBook seems to work for new books: 1/1
 - addBooks works with non-empty libraries: 1/1
 - getStatus works after adding books: 1/1

Let's try checking out books...
 - checkOut works when a book can be checked out: 1/1
 - checkOut works when a book is out of copies: 1/1
 - checkOut works when a book is not in the library: 1/1
 - getStatus works after checking out books: 1/1
```

```
Let's try returning books...
 - checkIn works when a book is not in the library: 1/1
 - checkIn works when a book is not checked out: 1/1
 - checkIn works when a book is checked back in: 1/1
 - getStatus works after checking out books: 1/1

Let's try to print the whole library...
 - toString works: 2/2

Let's ask the library some questions...
 - numBooksByAuthor works: 4/4
 - listBooksByAuthor works: 4/4
 - listBooksByTitle works: 4/4

Bonus! Let's remove a book from the library forever...
 - book deletion: 5/5

Total score: 35/30
   (can be up to 35 with bonus)
```

6 Project Submissions

To make sure you start early enough, this project is split into two submissions.

6.1 Part 1 – Author and Book Functionality

This part is to make sure your Author and Book implementations are functioning with enough time to implement the Library class afterwards. You won't be graded on following coding standards here, so if you're still missing a few javadoc comments, don't worry about it and just make sure you add them to the files before your Part 2 submission.

After submitting `Author.java` and `Book.java` to Part 1 on Canvas, you will be given access to the `LibraryTester.java` test file and the Part 2 assignment.

6.1.1 Turning in your assignment

Submit your `Author.java` and `Book.java` files to Canvas. Do not attach `.class` files or any other files.

6.1.2 Grading Rubric (total of 5 points)

5 points 0.1 points per point in score from `AuthorBookTester`

> Note: you may not get the full points for these tests if examination of your code indicates that you hard-coded answers to these specific test cases or otherwise did not actually implement the class as specified.

6.2 Part 2 – Full Project

This is the full project with your final versions of Author, Book, and Library. Even if you haven't changed your Author or Book implementations since part 1, make sure you submit all three files.

You will not be able to submit Part 2 without first submitting Part 1.

6.2.1 Turning in your assignment

Submit your `Author.java`, `Book.java`, and `Library.java` files to Canvas. Do not attach `.class` files or any other files.

6.2.2 Grading Rubric (total of 90 points + 10 bonus)

-5 points Files submitted to Canvas were not correctly named.

-5 points The code did not compile without errors or warnings.

-5 points Library implementation does not work with my version of Author and Book.

-5 points Member variables are not private.

-5 points The constructors and methods listed above are not public. (Private helper methods are fine, though.)

10 points The code adheres to the coding standard specified on the course website.

50 points (+ possibly 5 bonus) Score from author and book test in `LibraryTester`.

30 points (+ possibly 5 bonus) Score from running library test in `LibraryTester`.

Note: you may not get the full points for these tests if examination of your code indicates that you hard-coded answers to these specific test cases or otherwise did not actually implement the class as specified.