P03 BOOK LIBRARY

Posted on January 29, 2019 by Mouna AYARI BEN HADJ KACEM

- **SUBMIT your completed assignment by 9:59PM on Wednesday, February 13th 2019.** We will accept and grade submissions made through 9:59PM on Thursday, February 14th 2019 (HARD DEADLINE). But, NO CREDIT will be granted for any work that is submitted even one second after this hard deadline.
- **Pair Programming is allowed but not required for this assignment. Register your partnership no later than ~~Monday, February 11th~~ Tuesday, February 12th to work with a partner. If you have problems accessing this form, try following the advice here.**

CHANGELOG

Define one unit test method named testBookConstructorGetters() to check the constructor and the getter methods defined in your Book class.

IMPORTANT NOTES:

- A subscriber cannot return a book if it is not within its booksCheckedOut list. returnBook() method defined in the Subscriber class must check this condition. No specific error message has been specified in this write-up directions in case the subscriber tries to return a book which is not in his booksChechedOut list. So, it is up to you to display an error message or not for that case.
- If no match is found, findBookByTitle() and findBookByAuthor() methods defined in the Library class, return an empty ArrayList. This was specified in the javadocs by the comment detail "0 or more books can be found".

OVERVIEW

This program involves the design, implementation, and testing of a simplified version for a public library management system named Book Library. We distinguish two users for this application: (i) the subscriber who can borrow books, and (ii) the librarian who is responsible for managing the books in the library and helping the library subscribers to check out or return books. Further display options are also available.

OBJECTIVES AND GRADING CRITERIA

The goals of this assignment include:

- Practice how to create our own classes and use objects instances of these classes.
- Practice how to use ArrayList objects.
- Being more familiar with developing tests to check the good functioning of your code.

20	Online Tests: these automated grading test results are visible upon uploading your submission. You are allowed multiple opportunities to correct the organization and functionality of your code (if necessary).
20 points	Offline Tests: these automated grading tests are run after the assignment's deadline has passed. They check for similar functionality and organizational correctness as the Online Tests. Since you will not have opportunities to make corrections after seeing the feedback from these tests, you should consider and test the correctness of your own code as thoroughly as possible.
10 points	Code Readability: human graders will review the commenting, style, and organization of your final submission. They will be checking whether it conforms to the requirements of the CS300 Course Style Guide. Since you will not have opportunities to make corrections after seeing the feedback from these graders, you should consider and review the readability of your own code as thoroughly as possible.

SUBMISSION

For this assignment, you will need to submit 5 files through zybooks: Book.java, Subscriber.java, Librarian.java, Library.java, and BookLibraryTests.java You can make as many submissions as you would like prior to the assignment deadline.

APPLICATION DEMO

The following is a demo of your program. Note that the user's input below is shown in green, and that the rest of the text below was printed out by the program.

```
-------------------------------------------------------
     Welcome to our Book Library Management System
-------------------------------------------------------
Enter one of the following options:
[1 <password>] Login as a librarian
[2 <card bar code> <4-digits pin>] Login as a Subscriber
[3] Exit
-------------------------------------------------------
ENTER COMMAND: 1 abc

-------------------------------------------------------
    Welcome to Librarian's Space
-------------------------------------------------------
Enter one of the following options:
[1 <title> <author>] Add new Book
[2 <name> <pin> <address> <phone number>] Add new subscriber
[3 <card bar code> <book ID>] Check out a Book
[4 <card bar code> <book ID>] Return a Book for a subscriber
[5 <card bar code>] Display Personal Info of a Subscriber
[6 <card bar code>] Display Books Checked out by a Subscriber
[7] Display All Books
[8 <book ID>] Remove a Book
[9] Logout
-------------------------------------------------------
ENTER COMMAND: 1 Java Mouna
Book with Title Java is successfully added to the library.

-------------------------------------------------------
    Welcome to Librarian's Space
-------------------------------------------------------
Enter one of the following options:
[1 <title> <author>] Add new Book
[2 <name> <pin> <address> <phone number>] Add new subscriber
[3 <card bar code> <book ID>] Check out a Book
[4 <card bar code> <book ID>] Return a Book for a subscriber
[5 <card bar code>] Display Personal Info of a Subscriber
[6 <card bar code>] Display Books Checked out by a Subscriber
[7] Display All Books
[8 <book ID>] Remove a Book
[9] Logout
-------------------------------------------------------
ENTER COMMAND: 1 Python Gary
Book with Title Python is successfully added to the library.

-------------------------------------------------------
    Welcome to Librarian's Space
-------------------------------------------------------
Enter one of the following options:
[1 <title> <author>] Add new Book
[2 <name> <pin> <address> <phone number>] Add new subscriber
[3 <card bar code> <book ID>] Check out a Book
[4 <card bar code> <book ID>] Return a Book for a subscriber
[5 <card bar code>] Display Personal Info of a Subscriber
[6 <card bar code>] Display Books Checked out by a Subscriber
[7] Display All Books
[8 <book ID>] Remove a Book
[9] Logout
-------------------------------------------------------
ENTER COMMAND: 2 Ryan 1111 Madison 765432890
Library card with bar code 2019000001 is successfully issued to the new subscriber Ryan.

-------------------------------------------------------
    Welcome to Librarian's Space
-------------------------------------------------------
Enter one of the following options:
[1 <title> <author>] Add new Book
[2 <name> <pin> <address> <phone number>] Add new subscriber
[3 <card bar code> <book ID>] Check out a Book
[4 <card bar code> <book ID>] Return a Book for a subscriber
```

```
[5 <card bar code>] Display Personal Info of a Subscriber
[6 <card bar code>] Display Books Checked out by a Subscriber
[7] Display All Books
[8 <book ID>] Remove a Book
[9] Logout
---------------------------------------------------------
ENTER COMMAND: 5 2019000001
Personal information of the subscriber: 2019000001
  Name: Ryan
  Address: Madison
  Phone number: 765432890

---------------------------------------------------------
    Welcome to Librarian's Space
---------------------------------------------------------
Enter one of the following options:
[1 <title> <author>] Add new Book
[2 <name> <pin> <address> <phone number>] Add new subscriber
[3 <card bar code> <book ID>] Check out a Book
[4 <card bar code> <book ID>] Return a Book for a subscriber
[5 <card bar code>] Display Personal Info of a Subscriber
[6 <card bar code>] Display Books Checked out by a Subscriber
[7] Display All Books
[8 <book ID>] Remove a Book
[9] Logout
---------------------------------------------------------
ENTER COMMAND: 9

---------------------------------------------------------
     Welcome to our Book Library Management System
---------------------------------------------------------
Enter one of the following options:
[1 <password>] Login as a librarian
[2 <card bar code> <4-digits pin>] Login as a Subscriber
[3] Exit
---------------------------------------------------------
ENTER COMMAND: 2 2019000001 1111

---------------------------------------------------------
    Welcome to Subscriber's Space
---------------------------------------------------------
Enter one of the following options:
[1 <book ID>] Check out a book
[2 <book ID>] Return a book
[3 <title>] Search a Book by title
[4 <author>] Search a Book by author
[5] Print list of books checked out
[6] Print history of returned books
[7 <address>] Update address
[8 <phone number>] Update phone number
[9] Logout
---------------------------------------------------------
ENTER COMMAND: 4 Mouna
<Book ID>: 1 <Title>: Java <Author>: Mouna <Is Available>: true
---------------------------------------------------------
    Welcome to Subscriber's Space
---------------------------------------------------------
Enter one of the following options:
[1 <book ID>] Check out a book
[2 <book ID>] Return a book
[3 <title>] Search a Book by title
[4 <author>] Search a Book by author
[5] Print list of books checked out
[6] Print history of returned books
[7 <address>] Update address
[8 <phone number>] Update phone number
[9] Logout
---------------------------------------------------------
ENTER COMMAND: 1 1

---------------------------------------------------------
    Welcome to Subscriber's Space
---------------------------------------------------------
Enter one of the following options:
[1 <book ID>] Check out a book
[2 <book ID>] Return a book
[3 <title>] Search a Book by title
[4 <author>] Search a Book by author
[5] Print list of books checked out
[6] Print history of returned books
```

```
[7 <address>] Update address
[8 <phone number>] Update phone number
[9] Logout
------------------------------------------------------------
ENTER COMMAND: 5
Book ID: 1 Title: Java Author: Mouna

------------------------------------------------------------
    Welcome to Subscriber's Space
------------------------------------------------------------
Enter one of the following options:
[1 <book ID>] Check out a book
[2 <book ID>] Return a book
[3 <title>] Search a Book by title
[4 <author>] Search a Book by author
[5] Print list of books checked out
[6] Print history of returned books
[7 <address>] Update address
[8 <phone number>] Update phone number
[9] Logout
------------------------------------------------------------
ENTER COMMAND: 6
No books returned by this subscriber

------------------------------------------------------------
    Welcome to Subscriber's Space
------------------------------------------------------------
Enter one of the following options:
[1 <book ID>] Check out a book
[2 <book ID>] Return a book
[3 <title>] Search a Book by title
[4 <author>] Search a Book by author
[5] Print list of books checked out
[6] Print history of returned books
[7 <address>] Update address
[8 <phone number>] Update phone number
[9] Logout
------------------------------------------------------------
ENTER COMMAND: 9

------------------------------------------------------------
     Welcome to our Book Library Management System
------------------------------------------------------------
Enter one of the following options:
[1 <password>] Login as a librarian
[2 <card bar code> <4-digits pin>] Login as a Subscriber
[3] Exit
------------------------------------------------------------
ENTER COMMAND: 3

------------------------------------------------------------
       Thanks for Using our Book Library App!!!!
------------------------------------------------------------
```

STEP1. CREATE BOOK CLASS

Start by creating a new Java Project in eclipse and adding a new class called Book to this project. This class models a book within our book library. The specific fields that your Book class MUST include should be exactly the following:

```java
// class/static fields
private static int nextId = 1; // class variable that represents the identifier of the next
                               // book
// Instance fields
private final int ID; // unique identifier of this book
private String author; // name of the author of this book
private String title; // title of this book
private Integer borrowerCardBarCode; // card bar code of the borrower of this book
                                     // When borrowerCardBarCode == null, the book is available
                                     //                                    (no one has the book)
```

The description for the static and instance fields defined within the Book class is provided in the commented lines of code above.

Every instance of Book has a unique identifier of type int. These identifiers are incremented with the order of creation of a new Book. For instance, the first created book has 1 as ID, the second created book has 2 as ID, the third created book has 3 as ID, and so on.

You ARE NOT ALLOWED to add any additional field (instance or static) to this class.

A detailed JavaDoc description of the methods within your Book class **is provided within these Javadocs**. We note that it is OK to copy the JavaDocs for our own comments.

Make sure to implement the constructor and all of the methods according to their JavaDoc specifications. In particular, it is worth to note that when a new Book is created, it should be available. This means that its instance field <borrowerCardBarCode> must be set to null (no one has the book).

We note that you can add private helper methods to your classes in order to implement their specific behaviors. But, you are not allowed to add any public methods not specified in the provided **Javadocs**.

Make also sure to design and implement your own test methods to check that the constructor as well as the other methods including the accessors (getter methods) work correctly. To do so, create a new class called BookLibraryTests. All the test methods that you have to define for this program MUST be included within this class. All your test methods must be static and return a boolean.

In particular, you have to implement the following two test methods with exactly the following signatures:

- ```
  public static boolean testBookConstructorGetters(){}
  ```

testBookConstructorGetters() method checks whether the constructor of your Book class initializes correctly the new Book instance fields: title, author, borrowerCardBarCode, ID, and increments nextID static field. In particular, make sure that the different identifiers (ID fields) of at least two created books are set correctly and that the book is initially available. It checks also checks whether all your implemented getter methods defined within your Book class work correctly.

- ```
  public static boolean testBookReturnBook() {}
  ```

testBookReturnBook() checks whether returnBook() method defined within your Book class sets correctly the instance field borrowerCardBarCode. A Book must be available after this instance method is called.

STEP2. CREATE SUBSCRIBER CLASS

Now, create a new class called Subscriber. This class models a subscriber within our library management system. Each subscriber is issued a library card and has a PIN (personal information number) to access to the library services. We do not model the library card in this program. We rather define an instance field of type Integer called <CARD_BAR_CODE> that represents the bar code of the subscriber's library card. This card bar code is a 10-digits number. The issued card bar code numbers are incremented with accordance to the creation of the subscribers (2019000001, 2019000002, 2019000003, and so on). A subscriber can use this card bar code to have access to this application, borrow and return books from the library. He can also use this application to explore the library catalog (search for a book for instance).

The specific fields that your Subscriber class MUST include should be exactly the following:

```
// static fields
private final static int MAX_BOOKS_CHECKED_OUT = 10; // maximum number of books to be checked out
                                                     // one subscriber
private static int nextCardBarCode = 2019000001; // class variable that represents the card bar
                                                 // code of the next subscriber to be created
// Instance fields
private int pin; // 4-digits Personal Identification Number to verify the identity of this
                 // subscriber
private final Integer CARD_BAR_CODE; // card bar code of this subscriber

private String name; // name of this subscriber
private String address; // address of this subscriber
private String phoneNumber; // phone number of this subscriber

private ArrayList<Book> booksCheckedOut; // list of books checked out by this subscriber and not yet
                                         // returned. A subscriber can have at most 10 checked out books
private ArrayList<Book> booksReturned; // list of the books returned by this subscriber
```

The description for these fields is provided in the comments above. You ARE NOT ALLOWED to add any additional field (instance or static) to this class.

We further note that a subscriber cannot have more than MAX_BOOKS_CHECKED_OUT books checked out. He cannot also check out a book which is not available (already checked out).

Note that the JavaDoc description for **java.util.ArrayList** class is available **here**.

A detailed JavaDoc description of the methods within your Subscriber class **is provided within these Javadocs**. In particular, we provide you in the following with the implementation of the three display methods: printBooksCheckedOut(), printHistoryBooksReturned(), and printPersonalInfo(), and the implementation of isBookInBooksCheckedOut() method.

```java
/**
 * Checks if this subscriber booksCheckedOut list contains a given book
 * @param book book to check if it is within this subscriber booksCheckedOut list
 * @return true if booksCheckedOut contains book, false otherwise
 */
public boolean isBookInBooksCheckedOut(Book book) {
  return booksCheckedOut.contains(book);
}

/**
 * Displays the list of the books checked out and not yet returned
 */
public void displayBooksCheckedOut() {
  if (booksCheckedOut.isEmpty()) // empty list
    System.out.println("No books checked out by this subscriber");
  else
    // Traverse the list of books checked out by this subscriber and display its content
    for (int i = 0; i < booksCheckedOut.size(); i++) {
      System.out.print("Book ID: " + booksCheckedOut.get(i).getID() + " ");
      System.out.print("Title: " + booksCheckedOut.get(i).getTitle() + " ");
      System.out.println("Author: " + booksCheckedOut.get(i).getAuthor());
    }
}

/**
 * Displays the history of the returned books by this subscriber
 */
public void displayHistoryBooksReturned() {
  if (booksReturned.isEmpty()) // empty list
    System.out.println("No books returned by this subscriber");
  else
    // Traverse the list of borrowed books already returned by this subscriber and display its
    // content
    for (int i = 0; i < booksReturned.size(); i++) {
      System.out.print("Book ID: " + booksReturned.get(i).getID() + " ");
      System.out.print("Title: " + booksReturned.get(i).getTitle() + " ");
      System.out.println("Author: " + booksReturned.get(i).getAuthor());
    }
}

/**
 * Displays this subscriber's personal information
 */
public void displayPersonalInfo() {
  System.out.println("Personal information of the subscriber: " + CARD_BAR_CODE);
  System.out.println("  Name: " + name);
  System.out.println("  Address: " + address);
  System.out.println("  Phone number: " + phoneNumber);
}
```

Make sure to implement the constructor and all of the other methods according to their JavaDoc specifications. In addition, we note that your checkoutBook() method SHOULD display the following error messages:

- `"Checkout Failed: You cannot check out more than " + <MAX_BOOKS_CHECKED_OUT> + "books."`

  if the subscriber has already checked out the maximum number of books.

- `"You have already checked out " + <title of the book> + " book."`

  if the subscriber has already checked out that book.

- `"Sorry, " + <title of the book> + " is not available."`

  if the book is not available (already checked out by another subscriber).

Make sure also to design and implement your own test methods to check that the constructor as well as the other methods defined in this class including the accessors (getter methods) work correctly. In particular, you have to implement the following test method with exactly the following signature:

```java
public static boolean testSubscriberCheckoutBook() {}
```

This unit test method checks whether the checkoutBook() method defined within the Subscriber class works correctly. Hint: Make sure to check that the subscriber cannot have more than <MAX_BOOKS_CHECKED_OUT> books checked out and not yet returned, that the checked out book is added to the subscriber's <booksCheckedOut> list if it is available, and that the book's <borrowerCardBarCode> is correctly set if the book is successfully checked out by the subscriber. Hint: you can make call to isBookInBooksCheckedOut() from your test method to check if a book is added or not to the subscriber's booksCheckedOut list after calling checkoutBook() method.

STEP3. CREATE LIBRARIAN CLASS

Let's now create the Librarian class. This class models a librarian within our application. A librarian has a name and a password. These two fields should be defined as follows:

```
// instance fields
private String username; // librarian's username
private String password; // librarian password to have access to this application
```

A detailed description of the constructor and the different methods defined for the Librarian class **is provided within these Javadocs**. Make sure to implement all of them according to their JavaDoc specifications. You are also responsible for testing their good functioning.

Note also that your are not allowed to add any additional field (static or instance) to the Librarian class.

STEP4. CREATE LIBRARY CLASS

Now, create a new class called Library. This class models a simple book library. A Library object has only one Librarian, a street address, a list of all the books in the library, and a list of all its subscribers. These four instant fields should be defined as follows:

```
// instance fields
private String address; // Street address of this library
private Librarian librarian; // this library's librarian. This library must have only ONE
                             // librarian
private ArrayList<Book> books; // list of the books in this library
private ArrayList<Subscriber> subscribers; // list of this library's subscribers
```

You ARE NOT ALLOWED to add any additional field (instance or static) to this class.

A detailed description of the constructor and the different public methods defined for the Library class **is provided within these Javadocs**. Make sure to read carefully through the description of all these methods provided within the **Javadocs**. A few of them display specific messages. For instance,

- findBook(int bookId) should display the following message before returning if no book with the given bookId is found in the library books list:

  `"Error: this book identifier didn't match any of our books identifiers."`

- addBook(String title, String author) displays the following message after adding a new book with given title and author:

  `"Book with Title " + <title> + " is successfully added to the library."`

- addSubscriber(String name, int pin, String address, String phoneNumber) displays the following message after adding a new subscriber with the provided field values:

  `"Library card with bar code " + <card bar code> + " is successfully issued to the new subscriber " + <name> + "."`

- findSubscriber(int cardBarCode) displays the following error message if the provided cardBarCode did not match any of the subscribers' card bar codes:

  `"Error: this card bar code didn't match any of our records."`

At this stage, you have to implement and test the constructor of the class Library and its defined methods according to their Javadoc specification except main(), and readProcessUserCommand() methods. Directions on how to implement the latter ones are provided in the next step.

Note that you DO NOT have to check the validity of any input argument. We assume that they all have a correct format. For instance, you do not have to check if a <pin> is 4-digits number, if a <phone number> has a correct format, or if a provided <cardBarCode> is a 10-digits number not exceeding 2019999999.

In addition, we provide you in the following with the implementation of displayBooks() method. This static method displays the content of a given ArrayList of Book elements accordingly to a specific format.

```
/**
  * Displays a list of books
  *
  * @param books ArrayList of books
  */
 public static void displayBooks(ArrayList<Book> books) {
   // Traverse the list of books and display book id, title, author, and availability of each book
   for (int i = 0; i < books.size(); i++) {
     System.out.print("<Book ID>: " + books.get(i).getID() + " ");
```

```
      System.out.print("<Title>: " + books.get(i).getTitle() + " ");
      System.out.print("<Author>: " + books.get(i).getAuthor() + " ");
      System.out.println("<Is Available>: " + books.get(i).isAvailable());
    }
  }
```

You can notice that it would better to set the access modifier for the methods findSubscriber, addBook(), addSubscriber(), and removeBook() to **private**. Normally, these methods should not be visible and called from the outside of the class Library. But, we design them to be public methods in this program, so you can call them from your unit test methods defined in the BookLibraryTests class.

We would not also use the getters methods for the library address and the librarian. But, you may need them in your own test methods.

Note that you HAVE to implement a test method called testLibraryFindBookByAuthor() within your BookLibraryTests class. This method checks the good functioning of findBookByAuthor(String) method defined in the Library class. It should have exactly the following signature.

```
public static boolean testLibraryFindBookByAuthor() { }
```

Recall that your are responsible for defining additional test methods to check the good functioning of your methods. All in all, your submitted BookLibraryTests class MUST contain at least FIVE test methods.

STEP5. IMPLEMENT THE DRIVER METHOD FOR THE BOOK LIBRARY APPLICATION

Now, it is time to implement the driver method for our book library application. Two public methods are defined in the Library class for this purpose: the main() method and readProcessUserCommand(). We provide you in the following with how the main method of the Library class would be implemented. An example of the output of this main method is provided at the top of this write-up. NOTE that this method creates and uses only one instance of the Scanner class, and also only one instance of the Library class. We provide you also with an example of implementation of readProcessUserCommand() method. Feel free to use this code. Feel free also to organize your main method and break down readProcessUserCommand() the way that you prefer as long as you will have the same output.

```
  /**
   * Main method that represents the driver for this application
   *
   * @param args
   */
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in); // create a scanner object to read user inputs
    // create a new library object
    Library madisonLibrary = new Library("Madison, WI", "april", "abc");
    // read and process user command lines
    madisonLibrary.readProcessUserCommand(scanner);
    displayGoodByeLogoutMessage(); // display good bye message
    scanner.close();// close this scanner
  }
```

```
/**
   * Reads and processes the user commands with respect to the menu of this application
   *
   * @param scanner Scanner object used to read the user command lines
   */
  public void readProcessUserCommand(Scanner scanner) {
    final String promptCommandLine = "ENTER COMMAND: ";
    displayMainMenu(); // display the library management system main menu
    System.out.print(promptCommandLine);
    String command = scanner.nextLine(); // read user command line
    String[] commands = command.trim().split(" "); // split user command
    while (commands[0].trim().charAt(0) != '3') { // '3': Exit the application
      switch (commands[0].trim().charAt(0)) {
        case '1': // login as librarian commands[1]: password
          if (this.librarian.checkPassword(commands[1])) {
            // read and process librarian commands
            readProcessLibrarianCommand(scanner);
          } else { // error password
            System.out.println("Error: Password incorrect!");
          }
          break;
        case '2': // login as subscriber commands[1]: card bar code
          Subscriber subscriber = this.findSubscriber(Integer.parseInt(commands[1]));
          if (subscriber != null) {
            if (subscriber.getPin() == Integer.parseInt(commands[2])) // correct PIN
              // read and process subscriber commands
              readProcessSubscriberCommand(subscriber, scanner);
            else
              System.out.println("Error: Incorrect PIN.");
          }
          break;
      }
      // read and split next user command line
      displayMainMenu(); // display the library management system main menu
      System.out.print(promptCommandLine);
      command = scanner.nextLine(); // read user command line
      commands = command.trim().split(" "); // split user command line
    }
  }
```

In addition, we provide you with the following private helper methods that display the different application menus and the good bye message.

```java
/**
 * Displays the main menu for this book library application
 */
private static void displayMainMenu() {
  System.out.println("\n-------------------------------------------------------------");
  System.out.println("     Welcome to our Book Library Management System");
  System.out.println("-------------------------------------------------------------");
  System.out.println("Enter one of the following options:");
  System.out.println("[1 <password>] Login as a librarian");
  System.out.println("[2 <card bar code> <4-digits pin>] Login as a Subscriber");
  System.out.println("[3] Exit"); // Exit the application
  System.out.println("-------------------------------------------------------------");
}

/**
 * Displays the menu for a Subscriber
 */
private static void displaySubscriberMenu() {
  System.out.println("\n-------------------------------------------------------------");
  System.out.println("    Welcome to Subscriber's Space");
  System.out.println("-------------------------------------------------------------");
  System.out.println("Enter one of the following options:");
  System.out.println("[1 <book ID>] Check out a book");
  System.out.println("[2 <book ID>] Return a book");
  System.out.println("[3 <title>] Search a Book by title");
  System.out.println("[4 <author>] Search a Book by author");
  System.out.println("[5] Print list of books checked out");
  System.out.println("[6] Print history of returned books");
  System.out.println("[7 <address>] Update address");
  System.out.println("[8 <phone number>] Update phone number");
  System.out.println("[9] Logout");
  System.out.println("-------------------------------------------------------------");
}

/**
 * Displays the menu for the Librarian
 */
private static void displayLibrarianMenu() {
  System.out.println("\n-------------------------------------------------------------");
  System.out.println("    Welcome to Librarian's Space");
  System.out.println("-------------------------------------------------------------");
  System.out.println("Enter one of the following options:");
  System.out.println("[1 <title> <author>] Add new Book");
  System.out.println("[2 <name> <pin> <address> <phone number>] Add new subscriber");
  System.out.println("[3 <card bar code> <book ID>] Check out a Book");
  System.out.println("[4 <card bar code> <book ID>] Return a Book for a subscriber");
  System.out.println("[5 <card bar code>] Display Personal Info of a Subscriber");
  System.out.println("[6 <card bar code>] Display Books Checked out by a Subscriber");
  System.out.println("[7] Display All Books");
  System.out.println("[8 <book ID>] Remove a Book");
  System.out.println("[9] Logout");
  System.out.println("-------------------------------------------------------------");
}

/**
 * Display the Application GoodBye and logout message.
 */
private static void displayGoodByeLogoutMessage() {
  System.out.println("\n-------------------------------------------------------------");
  System.out.println("       Thanks for Using our Book Library App!!!!");
  System.out.println("-------------------------------------------------------------");
}
```

You can notice compile errors in your code. Two methods called in the provided code are not defined: readProcessLibrarianCommand() and readProcessSubscriberCommand(). These methods can be defined as private helper methods. They have the following signatures:

```java
/**
 * Reads and processes the librarian commands according to the librarian menu
 *
 * @param scanner Scanner object used to read the librarian command lines
 */
private void readProcessLibrarianCommand(Scanner scanner) {}

/**
 * Reads and processes the subscriber commands according to the subscriber menu
 *
 * @param subscriber Current logged in subscriber
 * @param scanner Scanner object used to read the librarian command lines
 */
private void readProcessSubscriberCommand(Subscriber subscriber, Scanner scanner) {}
```

The description of these methods is provided in the Javadoc comments above. We note that each of these methods should display the appropriate option menu for its user (either librarian or subscriber). And that each of these methods should prompt form multiple commands until the user logout, rather than one.

I would like also to note that you DO NOT have to check the validity of the provided user (librarian or subscriber) input commands. We assume that user command lines have a correct syntax and all the arguments have a correct format. For instance, you do not have to check if the number of arguments of each user command line is correct, if a <pin> is 4-digits number, if a <phone number> has a correct format, or if a provided <cardBarCode> is a 10-digits number. Your program may crash as a result of the user entering improperly formatted input. That's fine for this current version of the application. Whereas, you have to fix any NullPointerException that your code may throw while testing it.

We note also that this version of the book library considers book titles, addresses, and names consisting of only one word (not including white spaces). That's also fine.

The following methods from the Java8 API may be helpful while processing the user input command line:

- string.**equals**(Object anObject) – Compares this string to the specified object.
- string.**equalsIgnoreCase**(String anotherString) – Compares this string to anotherString, ignoring case considerations.
- string.**compareTo**(String anotherString) – Compares two strings lexicographically.
- string.**compareToIgnoreCase**(String str) – Compares two strings lexicographically, ignoring case differences.
- string.**charAt**(index) – Returns the character at the specified zero-based index from within this string.
- string.**toUpperCase**() – Returns a new copy of the string this method is called on, but with an upper case version of each alphabetic character.
- Integer.**parseInt**(String s): Parses the string argument as a signed decimal integer.

- White space is the command argument separator. Extra white spaces should be ignored while processing the user input command.
- String.**split**(): You can split the command line arguments using String.split(" ") and get the array of strings storing each argument of the command line in order.
- Finally, in order to avoid potential or solve Zybooks encoding problem that may occur when submitting your sourcecode files on zybooks, please refer to the following piazza note.

SUBMISSION

Congratulations on finishing this CS300 assignment! After verifying that your work is correct, and written clearly in a style that is consistent with the course style guide, you should submit your final work through zybooks. The most recent of your highest scoring submissions prior to the deadline of **9:59PM on Thursday February 14th (HARD DEADLINE)** will be recorded as your zybooks test score. NO CREDIT will be granted for any work that is submitted even one second after this hard deadline. The second portion of your grade for this assignment will be determined by running that same submission against additional automated grading tests after the submission deadline. Finally, the third portion of your grade for your P03 submission will be determined by humans looking for organization, clarity, commenting, and adherence to the course style guide.

EXTRA CHALLENGES

Here are some suggestions for interesting ways to extend this simulation, after you have completed, backed up, and submitted the graded portion of this assignment. **No extra credit will be awarded for implementing these features**, but they should provide you with some valuable practice and experience. DO NOT submit such extensions using zyBooks.

You can notice that the current application does not define a due date for the borrowed books.

- You can define a class Date that models a date associated with the hour (day, month, year, hour, minutes).
- You can add the following instance field to the Book class:
  - Date checkoutDate; // date when this book has been checked out. null if it is available
- Based on this information, you can extend your application such that the subscriber can get the list of the books overdue. He can also renew an overdue book.
- You can also extend your application such that a subscriber can hold a book which is checked out. When the book is returned, it will be held for that subscriber.