P04 – EXCEPTIONAL BOOK LIBRARY

Posted on February 9, 2019 by Mouna AYARI BEN HADJ KACEM

- **SUBMIT your completed assignment by 9:59PM on Wednesday, February 20th 2019.** We will accept and grade submissions made through 9:59PM on Thursday, February 21st 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 starting from Thursday February 14th and no later than Monday, February 18th to work with a partner. If you have problems accessing this form, try following the advice here.**

OVERVIEW

In this program, we are going to provide you with a source code for P3 developed by one of the CS300 teams. This code is similar to the one you have already submitted for P3. You may have noticed that your program developed so far for the Book Library application is not robust. It crashes in case of bad input provided by the user (for instance user command with incorrect syntax). We are going to handle all these erroneous input. We are also going to load books to the books Arraylist of a library from a file and save its content to a file too. Your program will also need to account for a variety of potential problems with the data in this file, and report appropriate warning messages for each.

OBJECTIVES AND GRADING CRITERIA

The goals of this assignment include:

- Learn how to improve the robustness of a program so it can survive unusual circumstances and cope with erroneous input without crashing.
- Develop your understanding of the difference between checked and unchecked exceptions, and get practice both throwing and catching exceptions of each kind.
- Get more practice writing tests, specifically tests that detect whether exceptions are thrown under the prescribed circumstances or not.

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).
30 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.

SUBMISSION

For this assignment, you will need to submit ONLY two files through zybooks: ExceptionalLibrary.java, and ExceptionalBookLibraryTests.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. It does not show the use of all the available options. But, it illustrates how erroneous input may be handled.

```
-------------------------------------------------------
     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
Syntax Error: Please enter a valid command!
 Arguments count is incorrect.

-------------------------------------------------------
     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 aaaa bbbb
Syntax error: The Subscriber bar code MUST be a 10-digits number.
 Argument number 1 within your command line is invalid.

-------------------------------------------------------
     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 aaaa
Error: Invalid PIN. The PIN MUST be a 4-digits number greater than or equal to 1000.
 Argument number 2 within your command line is invalid.

-------------------------------------------------------
     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 for a subscriber
[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
[L <filename.data>] Load list of Books from filename.data
[S <filename.data>] Save list of Books to filename.data
[9] Logout
-----------------------------------------------------------
ENTER COMMAND: 1
Syntax Error: Please enter a valid command!


-----------------------------------------------------------
    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 for a subscriber
[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
[L <filename.data>] Load list of Books from filename.data
[S <filename.data>] Save list of Books to filename.data
[9] Logout
-----------------------------------------------------------
ENTER COMMAND: 2 Mouna 123 Madison 54678907
Error: Invalid PIN. The PIN MUST be a 4-digits number greater than or equal to 1000.


-----------------------------------------------------------
    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 for a subscriber
[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
[L <filename.data>] Load list of Books from filename.data
[S <filename.data>] Save list of Books to filename.data
[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
```

```
[3] Exit
------------------------------------------------------
ENTER COMMAND: 3

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

STEP1. GETTING STARTED

While running your P3 Book Library program, you may have noticed that your program crashes when you enter a command line including a syntax error. We provide you in the following with two scenarios that illustrate such circumstances.

- For instance, if a command line defines 2 arguments (as its the case for "login as a librarian" command: [1 <password>] Login as a librarian) and the user enters this command line with only one argument (for instance 1 without providing any password), an ArrayIndexOutOfBoundsException will be thrown.

```
------------------------------------------------------
     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
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 1
        at ExceptionalLibrary.readProcessUserCommand(ExceptionalLibrary.java:   )
        at ExceptionalLibrary.main(ExceptionalLibrary.java:   )
```

- If your program is expecting to parse an Integer within a command line, but, the user enters instead an argument of different type (for instance a String), a NumberFormatException will be thrown.

```
------------------------------------------------------
     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 aaa bbb
Exception in thread "main" java.lang.NumberFormatException: For input string: "aaa"
        at java.lang.NumberFormatException.forInputString(Unknown Source)
        at java.lang.Integer.parseInt(Unknown Source)
        at java.lang.Integer.parseInt(Unknown Source)
        at ExceptionalLibrary.readProcessUserCommand(ExceptionalLibrary.java:   )
        at ExceptionalLibrary.main(ExceptionalLibrary.java:   )
```

In order to improve P3 program to cope with such erroneous input without crashing, other developers started refactoring their source code for P3 and came up with the Exceptional Book Library application. But, for time limitation, they have not been able to complete their source code appropriately.

We provide you in the following with the source code of their program. Your mission in this assignment is to complete the provided implementation with respect to the detailed Javadoc description provided within these **javadocs**.

FAMILIARIZING YOURSELF WITH EXCEPTIONAL BOOK LIBRARY

Start by downloading these four files: Book.java, Subscriber.java, Librarian.java, and ExceptionalLibrary.java. Add these files to a new project, and then create and add a fifth file named ExceptionalBookLibraryTests.java to the same project. To help familiarize yourself with this code, carefully read through it and add comments if necessarily as you go.

We recommend that you start with the Subscriber class. Notice that two final static fields have been added CARD_BAR_CODE_INIT and CARD_BAR_CODE_LAST. They define the range of new 10-digits card bar codes starting with 2019 that can be issued to the potential subscribers this year (2019). A new method named checkCardBarCode() has been also added to this class in order to check if a provided integer can be read as a card bar code that can be issued this year.

Now, pay close attention to the implementation of the constructor of the provided class Subscriber. If 999999 subscribers have been already added to the library, no new card bar codes can be issued. As a consequence, the registration operation (creating a new subscriber) will fail. To handle such circumstance, the constructor throws an InstantiationException including the following error message: “Error: CANNOT create a new subscriber. No more card can be issued.” Note that you do not need to change any thing within the Subscriber class and you won’t submit it through zybooks.

Read also through the implementation of Book and Librarian classes and add comments as needed as you go. You can compare your implementation of the three classes Book, Subscriber, and Librarian to the provided classes in the assignment.

Now, review ExceptionalLibrary class carefully while referring to the detailed methods documentation provided within these **javadocs**. With respect to P3, many new methods have been added to parse the user command lines and check their syntax, and validity of their arguments before running them. Some of them have complete implementation, others are missing (not yet implementation), and some include partial implementation dating from P3 (they do not cope with erroneous input). The latter ones include a //TODO tag.

Pay also attention that two new options have been added to the Librarian menu: Load a list of books and Save the list of books.

Read carefully through the provided code for the ExceptionalLibrary class. We are going to complete its implementation in the following steps. Note that to ensure encapsulation and implementation details hiding, it would better to define all the methods that parse and run the user command lines as private helper methods. But, we declare all these methods using the access modifier public to make them visible to our/your test methods.

While working on the next steps, you CANNOT add any instance or static field to any of the provided classes. You CANNOT add any public method not defined within these javadocs. You can add only private helper methods that you can call from the pre-defined public methods.

STEP2. HANDLE ERRONEOUS INPUT EXCEPTIONS

First add the following methods to your ExceptionalLibrary class and implement them according to their detailed javadocs description provided within these javadocs:

```
public int parsePinCode(String s, int errorOffset) throws ParseException {}
public int parseCardBarCode(String s, int errorOffset) throws ParseException {}
public int parseBookId(String s, int errorOffset) throws ParseException {}
```

Make sure also to consider the following details:

- PIN is valid if it can be parsed to an integer between 1000 and 9999, inclusive.
- A card bar code is valid if it can be parsed to an integer that passes Subscriber.checkCardBarCode() method.
- A bookId is considered valid is it can be parsed to an integer.

Note that Integer.parseInt(String) throws a NumberFormatException if the provided string does not contain a parsable integer. Whereas, all the above three methods MUST throw a ParseException (and not a NumberFormatException) if the provided String argument s cannot be parsed correctly. This means that if a NumberFormatException would be thrown, it should be caught by your methods parsePinCode, parseCardBarCode, and parseBookId. You can refer to the implementation of parsePhoneNumber() method provided for you in the ExceptionalLibrary class.

Here is an example on how to throw a new ParseException object:

```
throw new ParseException("error message", errorOffset);
```

The errorOffset is provided as input parameter for these methods.

Note that ParseException and InstantiationException are both checked exceptions defined by the Java API.

We note also that you can consider any error message that you find appropriate to describe the error as long as it CONTAINS the word "**error**". We won't be checking for an exact match with these error messages.

We recommend also that you remove //TODO tags from your code as you work on it. Your final submission on zybooks should not contain TODO comments or tags.

Make also sure to design and implement your own test methods to check that your three first parse methods work correctly. All the test methods that you have to define for this program MUST be included within ExceptionalBookLibraryTests class. All your test methods must be static and return a boolean.

In particular, you have to implement the following test method with exactly the following signature:

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

Now, after making sure that your parse argument methods work correctly, you have to complete the implementation of all the methods marked with the \\TODO tag with accordance to the provided javadocs. You have to check for erroneous input (check the number of arguments stored in the commands array, and parse its content).

- **Complete and update the implementation of parseRunLibrarianXXXCommand() methods**
  - parseRunLibrarianAddSubscriberCommand()
  - parseRunLibrarianCheckoutBookCommand()
  - parseRunLibrarianReturnBookCommand()
  - parseRunLibrarianDisplayPersonalInfoOfSubscriberCommand()
  - parseRunLibrarianDisplayBooksCheckedOutBySubscriberCommand()
  - parseRunLibrarianRemoveBookCommand()

IMPORTANT NOTES:

1. **DON'T CATCH** InstantiationException or ParseException if thrown in any of the parseRunLibrarianXXXCommand() methods. If thrown these two exceptions should propagate to the calling method.
2. **MAKE SURE** to check for the number of arguments within the commands input parameters (commands.length) with respect to the syntax provided for each command in the librarian menu. To do so, call checkCommandArgumentsCount() method.
3. **MAKE SURE** to parse the arguments stored in commands array if they represent a book identifier or a pin code or a card bar code, or a phone number.
4. The errorOffset of ParseException exception objects that would your parseRunLibrarianXXXCommand() methods throw, should be the index of the String command argument within the commands array, that was not parsable correctly. For instance, the content of the array commands that stores the command arguments for checking out a book for a subscriber is supposed to be as follows: {3, <card bar code>, <book id>}. So, the index of the command argument <card bar code> is 1 and the index of the command argument <book id> is 2. To parse <card bar code> within this commands array, the method parseCardBarCode(String s, int errorOffset) should be called as follows: parseCardBarCode(commands[1], 1).
5. If your parseRunLibrarianXXXCommand() methods should not throw unchecked exceptions such as IndexOutOfBoundsException or NullPointerException. These exceptions would refer to bugs in your program that should be fixed.
6. The order for checking the validity of each argument stored in commands array is equivalent to the order of these arguments within the array.
7. Make also sure to design and implement your own test methods to check that each of the above methods works correctly. In particular, you have to implement the following test method with exactly the following signature:

```
1 public static boolean testLibraryParseRunLibrarianCheckoutBookCommand() {}
```

- **Complete and update the implementation of parseRunSubscriberXXXCommand() methods**
  - parseRunSubscriberCheckoutBookCommand()
  - parseRunSubscriberReturnBookCommand()
  - parseRunSubscriberUpdatePhoneNumberCommand()

The SAME six first important notes provided for parseRunLibrarianXXXCommand() methods are also recalled for all the above parseRunSubscriberXXXCommand() methods.

Make also sure to design and implement your own test methods to check that each of the parseRunSubscriberXXXCommand() methods works correctly. In particular, you have to implement

the following test method with exactly the following signature:

```
1 public static boolean testLibraryParseRunSubscriberReturnBookCommand() {}
```

- **Complete and update the implementation of parseRunLoginAsLibrarian() and parseRunLoginAsSubscriber() methods**

Following the same logic, complete the implementation of both parseRunLoginAsLibrarian() and parseRunLoginAsSubscriber() with respect to their javadocs. These method should throw a ParseException if the number of the provided arguments within commands array is not as expected and if there is any invalid argument with respect to its format.

- **Complete and update the implementation of readProcessLibrarianCommand() method**

readProcessLibrarianCommand() method MUST CATCH any ParseException or InstantiationException that may be thrown from calling any of the parseRunLibrarianXXXCommand() methods. In response to catching these exceptions, the exception error message should be printed out to System.out.

At this stage, DO NOT implement options [L]oad a list of books and [S]ave the list of books from or to a filename file. Directions on how to appropriately implement these options will be provided in the next step.

You can run so far your program and check the output with respect to the implemented librarian menu options so far.

- **Complete and update the implementation of readProcessSubscriberCommand() method**

Following the same logic, readProcessSubscriberCommand() method MUST CATCH any ParseException or InstantiationException that may be thrown from calling any of the parseRunLibrarianXXXCommand() methods. In response to catching these exceptions, the exception error message should be printed out to System.out.

You can run so far your program and check the output with respect to the implemented librarian and subscriber menu options so far.

STEP3. LOAD AND SAVE BOOKS

Now, it is time to implement the new two options added to the Librarian menu:

```
[L <filename.data>] Load list of Books from filename.data
```

and

```
[S <filename.data>] Save list of Books to filename.data
```

These features should be implemented by **parseRunLibrarianLoadBooksCommand()** and **parseRunLibrarianSaveBooksCommand()** public methods conforming to their javadocs.

You have to check the number of arguments and format of every of these command lines provided by a librarian and throw an appropriate ParseException in case of any syntax error. Any error message that contains the word "error" will be accepted.

We note also that you are responsible for organizing the implementation of these load and save features into whatever helper methods you see fit. We recommend reading through the rest of this specification before you begin implementing any of these methods. We note also that, if needed, you should use the File.separator instead of '/' or '\' when specifying file paths so that your code will run on any platform. By default, the file path will be the path of your java project folder.

3.1. LOAD BOOKS

Now, we would like to implement a more interesting alternative that allows the librarian to add books to the library in an easy way. Instead of adding each book one by one given its title and author using add a new book command line option, the librarian can load a list of books saved in a text file. These books will be added as new books to the books ArrayList defined within the Library class. We would prefer files with extension (.data) or (.txt) or any other file type easily viewed and edited in Eclipse (or other text editors). Each line (terminated by a newline character) in these files may either be blank, or contains the title of a book, followed by ":", then followed by the author of the book. There may be additional spaces between the title of a book (or the author of a book) and the punctuation. All additional white spaces at the beginning and at the end of the title or author names must be deleted using Spring.trim() method while processing the file.

Here are a few examples of files that are conform to this specification books1.data, books2.data, and books3.txt.

Important note:

While processing the files, NullPointerException, or ArrayIndexOutOfBoundsException, or FileNotFoundException/IOException may be thrown if a line contains blank or only punctuation (":"), or no punctuation, or if the file cannot be found. Make sure to catch these exceptions within your methods defined to load a list of books from a file and do not propagate them to the calling method. If an ArrayIndexOutOfBoundsException or NullPointerException would be thrown while processing a file line, skip that line and continue processing the next line if any. Do not forget also to add a finally block to close open resources if an IOException such as FileNotFoundException would be thrown. You have to include descriptive and appropriate error messages that contain the word "error" to report an exception or an error while opening or processing the file.

We provide you in the following with an example of output using librarian load book option that you can take as reference.

```
-------------------------------------------------------
     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 for a subscriber
[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
[L <filename.data>] Load list of Books from filename.data
[S <filename.data>] Save list of Books to filename.data
[9] Logout
-----------------------------------------------------------
ENTER COMMAND: 1 Physics Ryan
Book with Title Physics 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 for a subscriber
[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
[L <filename.data>] Load list of Books from filename.data
[S <filename.data>] Save list of Books to filename.data
[9] Logout
-----------------------------------------------------------
ENTER COMMAND: 7
<Book ID>: 1 <Title>: Physics <Author>: Ryan <Is Available>: true

-----------------------------------------------------------
    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 for a subscriber
[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
[L <filename.data>] Load list of Books from filename.data
[S <filename.data>] Save list of Books to filename.data
[9] Logout
-----------------------------------------------------------
ENTER COMMAND: L books1.data

-----------------------------------------------------------
    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 for a subscriber
```

```
[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
[L <filename.data>] Load list of Books from filename.data
[S <filename.data>] Save list of Books to filename.data
[9] Logout
---------------------------------------------------------
ENTER COMMAND: 7
<Book ID>: 1 <Title>: Physics <Author>: Ryan <Is Available>: true
<Book ID>: 2 <Title>: java <Author>: mouna <Is Available>: true
<Book ID>: 3 <Title>: Data Structures <Author>: mouna <Is Available>: true
<Book ID>: 4 <Title>: Object Oriented Programming <Author>: Gary <Is Available>: true
<Book ID>: 5 <Title>: java <Author>: mouna <Is Available>: true
<Book ID>: 6 <Title>: c# <Author>: gary <Is Available>: true

---------------------------------------------------------
    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 for a subscriber
[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
[L <filename.data>] Load list of Books from filename.data
[S <filename.data>] Save list of Books to filename.data
[9] Logout
---------------------------------------------------------
ENTER COMMAND: l books2.data

---------------------------------------------------------
    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 for a subscriber
[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
[L <filename.data>] Load list of Books from filename.data
[S <filename.data>] Save list of Books to filename.data
[9] Logout
---------------------------------------------------------
ENTER COMMAND: 7
<Book ID>: 1 <Title>: Physics <Author>: Ryan <Is Available>: true
<Book ID>: 2 <Title>: java <Author>: mouna <Is Available>: true
<Book ID>: 3 <Title>: Data Structures <Author>: mouna <Is Available>: true
<Book ID>: 4 <Title>: Object Oriented Programming <Author>: Gary <Is Available>: true
<Book ID>: 5 <Title>: java <Author>: mouna <Is Available>: true
<Book ID>: 6 <Title>: c# <Author>: gary <Is Available>: true
<Book ID>: 7 <Title>: java <Author>: Stephanie <Is Available>: true
<Book ID>: 8 <Title>: Mathematics <Author>: Marc <Is Available>: true
<Book ID>: 9 <Title>: Data Structures and Algorithms <Author>: Michael <Is Available>: t
<Book ID>: 10 <Title>: Object Oriented Programming <Author>: Gary <Is Available>: true
<Book ID>: 11 <Title>: Python <Author>: mouna <Is Available>: true

---------------------------------------------------------
    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 for a subscriber
[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
[L <filename.data>] Load list of Books from filename.data
[S <filename.data>] Save list of Books to filename.data
[9] Logout
-----------------------------------------------------------
ENTER COMMAND: L
Syntax Error: Please enter a valid command!


-----------------------------------------------------------
    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 for a subscriber
[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
[L <filename.data>] Load list of Books from filename.data
[S <filename.data>] Save list of Books to filename.data
[9] Logout
-----------------------------------------------------------
ENTER COMMAND: L books3.data
Error: Could NOT load books contents from file books3.data

-----------------------------------------------------------
    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 for a subscriber
[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
[L <filename.data>] Load list of Books from filename.data
[S <filename.data>] Save list of Books to filename.data
[9] Logout
-----------------------------------------------------------
ENTER COMMAND: L books3.txt
Error: Found incorrectly formatted line in file books3.txt: :

-----------------------------------------------------------
    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 for a subscriber
[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
[L <filename.data>] Load list of Books from filename.data
[S <filename.data>] Save list of Books to filename.data
[9] Logout
---------------------------------------------------------
ENTER COMMAND: 7
<Book ID>: 1 <Title>: Physics <Author>: Ryan <Is Available>: true
<Book ID>: 2 <Title>: java <Author>: mouna <Is Available>: true
<Book ID>: 3 <Title>: Data Structures <Author>: mouna <Is Available>: true
<Book ID>: 4 <Title>: Object Oriented Programming <Author>: Gary <Is Available>: true
<Book ID>: 5 <Title>: java <Author>: mouna <Is Available>: true
<Book ID>: 6 <Title>: c# <Author>: gary <Is Available>: true
<Book ID>: 7 <Title>: java <Author>: Stephanie <Is Available>: true
<Book ID>: 8 <Title>: Mathematics <Author>: Marc <Is Available>: true
<Book ID>: 9 <Title>: Data Structures and Algorithms <Author>: Michael <Is Available>: t
<Book ID>: 10 <Title>: Object Oriented Programming <Author>: Gary <Is Available>: true
<Book ID>: 11 <Title>: Python <Author>: mouna <Is Available>: true
<Book ID>: 12 <Title>: Algebra <Author>: Goss <Is Available>: true
<Book ID>: 13 <Title>: Music <Author>: Beethoven <Is Available>: true
<Book ID>: 14 <Title>: Python <Author>: Gary <Is Available>: true

---------------------------------------------------------
    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 for a subscriber
[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
[L <filename.data>] Load list of Books from filename.data
[S <filename.data>] Save list of Books to filename.data
[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!!!!
---------------------------------------------------------
```

3.2. SAVE BOOKS

Finally, let's implement parseRunLibrarianSaveBooksCommand() method. We would like to save the content of books ArrayList defined in the ExceptionalLibrary class in a file conforming to the following format for each line:

```
1 title:author
```

Do not save the books identifiers, or any information about their availability (for instance borrowerCardBarCode). You should be able to read the saved file from Eclipse or any text

editor. You can also load a saved file. The list of books must be added as new books to the ExceptionalLibrary ArrayList books. This list can contain books with the same title and author, but each book should have a different (unique) identifier.

We recommend the use of **PrintWriter** or **PrintStream** classes defined in the java.io package to write and save a file.

Make also sure to design and implement your own test methods to check the good functioning of the implemented methods. All in all, your submitted ExceptionalBookLibraryTests class MUST contain at least FIVE test methods.

CODE REVIEW

After implementing and testing your Exceptional Book Library program, you should perform a code review of this code. Make sure to remove all the //TODO tags from your code before its final submission.

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 21st (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. Even though P04 won't be graded by humans, make sure that your program's organization, clarity, commenting is conform 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 consider to the same suggestions provided for P3 in P4 by adding checkoutDate and returnDate to the Book class and suggest a way to handle overdue books.
- Define a format for the phone number, for instance (xxx)-xxx-xxxx and check the validity of the provided phone numbers accordingly.
- Before adding a new subscriber, you can check if it already exists in the subscribers list considering its phoneNumber as search key.
- You can save Book.nextId and Subscriber.nextCardBarCode static fields in a textfile when the user quits/closes the application. They would be loaded when the application restarts. That way, you can save the identifiers of each book and its availability and not only title and authors fields. The saved file can be loaded when the application starts if it is found.
- You can define a way to save the list of subscribers before quitting the application and load it when the application starts if found. You do not need to save the list of subscribers returned

books. This history would be cleaned up each time the application turns off. You can restore the list of books checked out by exploring and traversing the list of saved books after loading it.