Library Project: Reader.java

In the project, you will develop four classes called Book, **Reader**, Shelf, and Library. These classes will be used to store, manipulate, and analyze information about the objects stored in a library.

This details the creation of the Reader.java class. The UML is listed below.

All the fields have getters and setters. It is also important to note that the class has a hashCode() method and an equals() method.

Field Details

Constant Fields

CARD_NUMBER_
NAME_
PHONE_
BOOK_COUNT_
BOOK_START_

These fields will hold the index of the values shown below in a String array.
Note that BOOK_COUNT_ will be used to process books the reader has checked out.
BOOK_START_ will be the beginning of a listing of books the reader has checked out.

Library00.csv (reader values listed in bold)

1,Drew Clinkenbeard,831-582-4007,2,42-w-87,2020-10-12,1337,2020-11-1
2,Jennifer Clinkenbeard,831-555-6284,1,42-w-87,2020-05-05
3,Monte Ray,555-555-4444,2,42-w-87,2020-12-12,1337,2021-1-2
4,Laurence Fishburn,831-582-4007,2,42-w-87,2019-02-18,1337,2025-10-10

Method Details

addBook(Book book)

If the Reader already has a copy of book addBook returns a `Code.BOOK_ALREADY_CHECKED_OUT_ERROR;`

Otherwise add the book to the List of books in the reader object and return a `Code.SUCCESS;`

removeBook(Book book)

If the reader doesn't have the book in their possession removeBook should return a `Code.READER_DOESNT_HAVE_BOOK_ERROR`

If the book is successfully removed from the list of Books then a Code.SUCCESS should be returned.

For any other issue a `Code.READER_COULD_NOT_REMOVE_BOOK_ERROR` should be returned.

hasBook(Book book)

Returns true if the user has the book in their list

equals() and hashCode()

Compares all the fields except the list of books.

toString()

Returns a string that looks like:
Bob Barker (**#2187**) has checked out {**Book1**, **Book2**}

Where **Bob Barker** is the name stored in the name field, **2187** is the card number, and the books listed are all the books in their inventory.

Testing

This class should be tested in the same way as Book.java with the addition of tests for the methods addBook, removeBook, and hasBook.

addBook_test

To test this class the Book class will need to be completed.

1. Add a Book to the Reader and check the status code.
2. The first time the book is added an `assertEquals(reader.addBook(book), Code.SUCCESS)` will pass
3. A second attempt to add the book will result in assertNotEquals passing when compared to a Code.Success
4. A second attempt to add the book will result in assertEquals passing when compared to `Code.BOOK_ALREADY_CHECKED_OUT_ERROR;`

getBookCount_Test

To test this class the Book class will need to be completed.

1. Create a Reader object.
    a. Assert that book count is 0
2. Add a Book to the Reader object
    a. Assert that book count is 1
3. Remove a Book from the Reader
    a. Assert that bookCount is 0

hasBook_test

1. Create a Book
2. Use hasBook to check if the Reader has the book, should return false
    a. assertFalse should pass
3. Use addBook to add the book to the Reader
4. hasBook should now return true
    a. assertTrue should pass

removeBook_test

1. Ensure that assertEquals passes when compared to `Code.READER_DOESNT_HAVE_BOOK_ERROR` before a Book is added
2. Add a book

Ensure that assertEquals passes when compared to when Code.SUCCESS removing the book

UML Diagram of Reader.java