Library Project: Library.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 Library.java class. The UML is listed below.

All the fields have getters and setters.

It is also important to note that the class has neither a hashCode() method nor an equals() method.

Field Details

Constant Fields

LENDING_LIMIT
This is the maximum number of books someone can check out at a time. Initially set to 5.

Members

String name

The name of the library

Static int libraryCard

The current max library card number

List<Reader> readers

A list of readers registered to the library

HashMap<String, Shelf> shelves

A HashMap containing the subject (String) of the shelf and the Shelf object

HashMap<Book, Integer> books

A HashMap that contains the Book objects registered to the library and the count of those books (Integer)

Method Details

init(String filename)

This is the method that is responsible for parsing the Library00.csv file into a collection of objects.
The parameter filename is the name of the file to be parsed (in this case Library00.csv).
The init method will call other methods from this class to aid in the processing of the csv file.

The csv files will always have a structure like that shown in below. The csv file will contain a list of comma separated values that represent books, shelves, and readers respectively. The numbers 9,3,and 4 all correspond to the number of records to follow. Each of these will be parsed by their respective init methods (initBooks, initShelves, and initReader respectively)

If the filename can't be found, return a `Code.FILE_NOT_FOUND_ERROR`.

Use the convertInt method to return the number of books to parse. Details of the covertInt method below.

If the number returned by parseInt is less than 0 (corresponding to an error code) return the error code related to the number (Enums may be iterated). If there is no corresponding error return a
`Code.UNKNOWN_ERROR`
It may be wise to make a method for this.

After the books have been parsed, call the listBooks() method.

Follow the same procedure for shelf and reader, listing the parsed shelves and the parsed readers after each call to their respective 'init' methods. Returning any related error codes.

Library00.csv

```
9
1337,Headfirst Java,education,1337,Grady Booch,0000
42-w-87,Hitchhikers Guide To the Galaxy,sci-fi,42,Douglas Adams,0000
42-w-87,Hitchhikers Guide To the Galaxy,sci-fi,42,Douglas Adams,0000
1337,Headfirst Java,education,1337,Grady Booch,0000
42-w-87,Hitchhikers Guide To the Galaxy,sci-fi,42,Douglas Adams,0000
5297,Count of Monte Cristo,Adventure,999,Alexandre Dumas,0000
42-w-87,Hitchhikers Guide To the Galaxy,sci-fi,42,Douglas Adams,0000
1337,Headfirst Java,education,1337,Grady Booch,0000
34-w-34,Dune,sci-fi,235,Frank Herbert,0000
3
1,sci-fi
2,education
3,Adventure
4
1,Drew Clinkenbeard,831-582-4007,2,42-w-87,2020-10-12,1337,2020-11-01
2,Jennifer Clinkenbeard,831-555-6284,1,42-w-87,2020-05-05
3,Monte Ray,555-555-4444,2,42-w-87,2020-12-12,1337,2021-01-02
4,Laurence Fishburn,831-582-4007,2,42-w-87,2019-02-18,1337,2025-10-10
```

initBooks(int bookCount, Scanner scan)

This method takes an integer, the number of books to parse, and a Scanner.
The scanner represents the current position of the CSV file that is being parsed.

The int bookCount represents the number of books to parse.
If bookCount is less than 1, return a `Code.LIBRARY_ERROR`
(this should never happen but let's be defensive.)

Iterate through the records in the CSV file converting each line to a book. Use the constant fields in the Book object to identify the indexes of the relevant data from the CSV file.

Use convertInt to parse the page number. If the page number is 0 or less, return a `Code.PAGE_COUNT_ERROR`

Use the converDate method described below to parse the date. If the date comes back as a null return a `Code.DATE_CONVERSION_ERROR`

Finally call the addBook() method to add the book to the list of books in the Library object.

Return a `Code.SUCCESS`

initShelves(int shelfCount, Scanner scan)

This method works very similarly to initBooks. The int shelfCount is used to determine how many shelves to parse out of the Scanner object.

If shelfCount is less than 1 return a `Code.SHELF_COUNT_ERROR`

If the number for the shelf does not parse correctly, return a `Code.SHELF_NUMBER_PARSE_ERROR`

Use the `addShelf()` method to add the shelves to the HashMap shelves in the Library object.

Once all the shelves have been parsed, use the `addBookToShelf()` method to add all the books from the List of books from the current Library object to the corresponding shelves.[1]

Finally check that the size of the shelves object matches the value for shelf count, if it does return a Code.SUCCESS. If it does not print a message that reads
`"Number of shelves doesn't match expected"`
And return a `Code.SHELF_NUMBER_PARSE_ERROR`

[1] This is duplicated in addShelf. So this introduces a bug but won't be a source of lost points.

initReader(int readerCount, Scanner scan)

Works in the same way as initBooks and initShelves. The int is used to determine how many readers to parse and the scanner contains the current text file.

If the reader count is less than or equal to 0 return a `Code.READER_COUNT_ERROR`

Add each reader to the list of readers (use the List.add() method)

The thing that makes initReader tricky is that each reader may have a list of books. The number of books to parse is indexed by Reader.BOOK_COUNT_ and the starting position of the first book is stored in Reader.BOOK_START_

Find the books in the Library object using the getBookByISBN() method.

If a book does not exist in the library print out the string "`ERROR`" and continue to parse the remaining books.

Use the convertDate method to convert the due date to a LocalDate object.

Once the book is found, use the checkOutBook method to add the book to the reader.

When complete return a `Code.SUCCESS`

addbook(Book newBook)

If the HashMap of books in the Library object contains the book, increment the count and print out a String like:
`"3 copies of [Book.title] in the stacks"`
Where 3 is the number of copies and Best Served Cold is the title of the book that has been added.

If the book does NOT exist add the book to the HashMap of books setting the count to 1.
Print out a like like:
`"[Book.title] added to the stacks."`
Where Best Served Cold is the title of the book that has been added.

If there is a shelf with a matching subject, add the book to that shelf and return a `Code.SUCCESS`

Otherwise print out
`"No shelf for [subject] books"`
Where Adventure is the subject of the book and return a `Code.SHELF_EXISTS_ERROR`

returnBook(Reader reader, Book book)

If the reader does not have the book in their List print a message like
`"[Reader.name] doesn't have [book title] checked out"`
And return a `Code.READER_DOESNT_HAVE_BOOK_ERROR`

Print out
`"[Reader.getName] is returning [book]"`
Use the Reader.removeBook(book) method to remove the book and store the Code that is returned.

If the code is a `Code.SUCCESS` then call the returnBook(book) method and return the resulting code.

If the code is NOT a success print
`"Could not return [Book]"`

And return the Code object

returnBook(Book book)

Check if there is a shelf with a matching subject in the HashMap of shelves, if not print out
`"No shelf for [book]"`
And return a `Code.SHELF_EXISTS_ERROR`
Otherwise call the shelf.addBook(book) method for the appropriate shelf from the HashMap stored in the library object.

addBookToShelf(Book book, Shelf shelf)

Try to return the book using the returnBook(book) method, if the returnBook method returns a `Code.SUCCESS` then return a `Code.SUCCESS` and exit.

If the shelf that was supplied does not match the subject of the book supplied return a `Code.SHELF_SUBJEC_MISMATCH_ERROR`

Use the shelf.addBook(book) method to add the book to the shelf. If it is successful print
`"[book] added to shelf"`
And return a `Code.SUCCESS`

Otherwise print
`"Could not add [book] to shelf"`

And return whatever code was returned from the shelf.addBook(book) method.

listBooks()

List all the books at the library, even those not on shelves. THe print out should look like:

```
4 copies of Hitchhikers Guide To the Galaxy by Douglas Adams ISBN:42-w-87
3 copies of Headfirst Java by Grady Booch ISBN:1337
1 copies of Dune by Frank Herbert ISBN:34-w-34
1 copies of Count of Monte Cristo by Alexandre Dumas ISBN:5297
```

Return the total count of all books at the library. For the list above it would return 9 (4+3+1+1)

checkoutBook(Reader reader, Book book)

Check for the reader, if no reader print out
`"[name] doesn't have an account here"`
and return a `Code.READER_NOT_IN_LIBRARY_ERROR`

Check the reader has not exceeded the lending limit, if they have print out
`"[NAME] has reached the lending limit, ([LENDING_LIMIT_])"`
And return a `Code.BOOK_LIMIT_REACHED_ERROR`

Check for the book in the HashMap books in the Library object, if it isn't there print out
`"ERROR: could not find [BOOK]"`
and return a `Code.BOOK_NOT_IN_INVENTORY_ERROR`

Check for a shelf for the book, if no shelf exists print out
`"no shelf for [subject] books!"`
return a `Code.SHELF_EXISTS_ERROR`

If the Shelf has less than 1 copy of the book print out
`"ERROR: no copies of [book] remain"`
And return a `Code.BOOK_NOT_IN_INVENTORY_ERROR`

Use the reader.addBook() method if it does not return a `Code.SUCCESS`
Print out
`"Couldn't checkout [book]"`
and return the code that was returned from the addBook method

Call shelf.removeBook(book)
If the code from removeBook is a success print
`"[Book] checked out successfully"`
Return the code

getBookByISBN(String isbn)

Returns a Book object from the HashMap of books in the Library object that matches the book the String isbn. If no such object exists, print out

```
"ERROR: Could not find a book with isbn: [isbn]"
```

And return null

listShelves(boolean showbooks)

If the boolean showbooks is true, call the 'listbooks' method for each shelf.
If the boolean showbooks is false display the toString of the each shelf object
Return a `Code.SUCCESS`

addShelf(String shelfSubject)

Create a Shelf object assign a shelf number of the size of the shelves HashMap plus one and a subject of shelfSubject. Call `addShelf(shelf)` with that object.

addShelf(Shelf shelf)

If a shelf already exists in the shelves HashMap print out

```
"ERROR: Shelf already exists [shelf]"
```

And return a `Code.SHELF_EXISTS_ERROR`

Add the shelf to the HashMap of shelves and assign it the next shelf number.
The next shelf number will be the largest number in the HashMap plus one.

Add all the books with matching subjects to the new shelf.[2]

Return a `Code.SUCCESS`

getShelf(Integer shelfNumber)

Return the Shelf from the HashMap shelves that matches the shelfNumber.
If no Shelf matches the number print out

```
"No shelf number [shelfNumber] found"
```

And return a null

getShelf(String subject)

Return the Shelf from the HashMap shelves that matches the subject.
If no Shelf matches the number print out

```
"No shelf for [subject] books"
```

And return a null

---

[2] This introduces a bug. If you have this when it is turned in, that is fine, but this line that is the source of the bug that made Books be double counted. This does mean that the Util.java will need to be updated…

<u>**listReaders()**</u>

Print out the toString of all the readers in the List of readers
Return the total number of readers

listReaders(boolean showBooks)

If showBooks is true, display the reader and all their books in a display like:

```
Drew Clinkenbeard(#1)  has the following books:
[Hitchhikers Guide To the Galaxy by Douglas Adams ISBN:42-w-87, Headfirst
Java by Grady Booch ISBN:1337]
Jennifer Clinkenbeard(#2)  has the following books:
[Hitchhikers Guide To the Galaxy by Douglas Adams ISBN:42-w-87]
Monte Ray(#3)  has the following books:
[Hitchhikers Guide To the Galaxy by Douglas Adams ISBN:42-w-87, Headfirst
Java by Grady Booch ISBN:1337]
Laurence Fishburn(#4)  has the following books:
[Hitchhikers Guide To the Galaxy by Douglas Adams ISBN:42-w-87, Headfirst
Java by Grady Booch ISBN:1337]
```

And return the total number of readers

If showBooks is false print out the toString of all the readers in the HashMap of readers
Return the total number of readers

getReaderByCard(int cardNumber)

Returns a reader whose library card number matches cardNumber

If no such reader exists print "Could not find a reader with card #[cardNumber]"
And return null

addReader(Reader reader)

Adds reader to the List of readers in the Library object. This method will require a new Code be added to the Code object.

`READER_ALREADY_EXISTS_ERROR(-47, "Reader already exists!")`

If there is alreader a reader object that equals reader, print
`"[reader name] already has an account!"`
And return a `Code.READER_ALREADY_EXISTS_ERROR`

If there is another reader with the same card number, print
`"[existing reader name] and [reader.name] have the same card number!"`
And return a `Code.READER_CARD_NUMBER_ERROR`

If the reader is successfully added print
`"[reader.name] added to the library!"`
If the reader object's library card value is larger than the current field libraryCard in the Library object, set the libraryCard field to that value.
Lastly return a `Code.SUCCESS`

removeReader(Reader reader)

Removes a reader from the List of readers in the Library object. This method will require a new Code to be added to the Code object.

```
READER_STILL_HAS_BOOKS_ERROR(-48, "Must return all books.")
```

If the reader exists in the List of readers but they have books in their inventory, print
`"[reader name] must return all books!"`
And return a `Code.READER_STILL_HAS_BOOKS_ERROR;`

If the reader is not in the List of Readers in the Library object print
"[reader]
 is not part of this Library"
And return a `Code.READER_NOT_IN_LIBRARY_ERROR`

If the reader exists in the Library object and has no books in their inventory, remove them from the List of readers and return a `Code.Success`

convertInt(String recordCountString, Code code)

Returns the Integer form of the recordCountString.
If there is a NumberFormateException when converting recordCountString to an integer, a different message is printed depending on the Code that was supplied.

If there is a NumberFormateException the following is printed
`"Value which caused the error: [recordCountString]"`
`"Error message: [Code.message]"`
Return the numerical code associated with the Code object.
Additionally, depending on the Code that was supplied as a parameter, different messages are printed.
See the table below for which message to print.

Code Supplied	Message printed
`Code.BOOK_COUNT_ERROR`	`"Error: Could not read number of books"`
`Code.PAGE_COUNT_ERROR`	`"Error: could not parse page count"`
`Code.DATE_CONVERSION_ERROR`	`"Error: Could not parse date component"`
Any other code	`"Error: Unknown conversion error"`

If there is **not** an error return the Integer form of recordCountString

convertDate(String date, Code errorCode)

Converts the date String to a LocalDate object. The date String is expected to be in the format:
```
"yyyy-mm-dd"
```
For example for February, 18 1982 the String would look like:
```
"1982-02-18"
```

If the date string is "0000" return a LocalDate object set to 01-Jan-1970

If the date String does not split into 3 elements on a "-" characters print out
```
"ERROR: date conversion error, could not parse [date]"
"Using default date (01-jan-1970)"
```
And return a LocalDate object set to 01-Jan-1970

If any of the converted values from the split String are less than 0 print out :

```
"Error converting date: Year [year] "
"Error converting date: Month [month]"
"Error converting date: Dat [day]"
"Using default date (01-jan-1970"
```
Where [year], [month], and [day] are the converted values,
and return a LocalDate object set to 01-Jan-1970

If there are no errors return a LocalDate object set to the parsed date values.

getLibraryCardNumber()

Returns the current value for libraryCard +1

errorCode(int codeNumber)

Returns the Code object associated with codeNumber
If there is no matching number return a
`Code.UNKNOWN_ERROR`

Testing Library.java

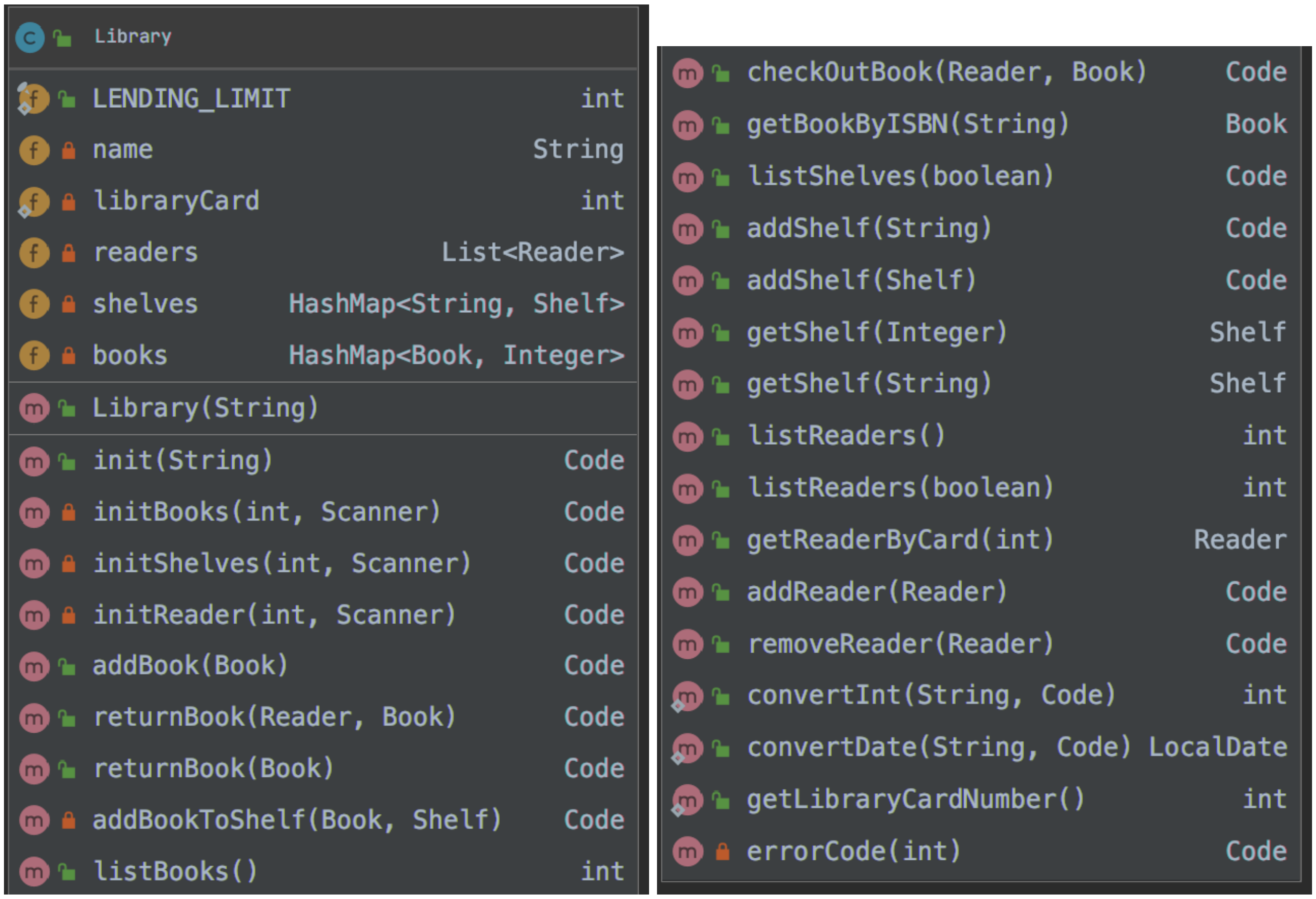
Test this like we have tested everything else.
Ensure that we can create a library.
Ensure that each method returns the expected value. If it returns a code, ensure it is the correct code. If it is a number, ensure it is the correct number.

If it returns an object from this project (Book, Shelf, or Reader) ensure that it is the CORRECT object.
If you ask for shelf 1, make sure it is shelf 1.

For overloaded methods, test them all but only test the returned value. Specifically `listReader()` and `listReader(Boolean)`. Ideally one should be calling the other (listReader() should call `listReader(Boolean)` with a `false` parameter).

A good way to ensure everything is tested is to develop the tests along with the methods.

UML Diagram of Library.java (Link to full size image[3])



```
Library
LENDING_LIMIT                          int
name                                String
libraryCard                            int
readers                       List<Reader>
shelves          HashMap<String, Shelf>
books            HashMap<Book, Integer>
Library(String)
init(String)                          Code
initBooks(int, Scanner)               Code
initShelves(int, Scanner)             Code
initReader(int, Scanner)              Code
addBook(Book)                         Code
returnBook(Reader, Book)              Code
returnBook(Book)                      Code
addBookToShelf(Book, Shelf)           Code
listBooks()                            int
```

```
checkOutBook(Reader, Book)            Code
getBookByISBN(String)                 Book
listShelves(boolean)                  Code
addShelf(String)                      Code
addShelf(Shelf)                       Code
getShelf(Integer)                    Shelf
getShelf(String)                     Shelf
listReaders()                          int
listReaders(boolean)                   int
getReaderByCard(int)                Reader
addReader(Reader)                     Code
removeReader(Reader)                  Code
convertInt(String, Code)               int
convertDate(String, Code) LocalDate
getLibraryCardNumber()                 int
errorCode(int)                        Code
```

[3] https://drive.google.com/file/d/1NGf_SzxrSxVu3cKFHmLNJt0tHC3Lh3p0/