Seneca College

Feb 17, 2017

SCHOOL OF INFORMATION AND COMMUNICATIONS TECHNOLOGY

JAC444 **Due date: March 31/17**

Assignments 2 & 3

1. Goals

The goals of these assignments are: (1) develop and implement Java classes for a business application using specific data structures and algorithms; (2) write code for testing the soundness and completeness of your solution.

2. Tasks

Your project is to design and develop an inventory system for keeping track of books borrowed from libraries in a college.

We start with the assumptions: (1) libraries have unique names and, (2) a library can stockpile a variable number of books. Furthermore, a book has a unique name and an internal tag significant for the library inventories (an integer with values between -100 and 100). These two values, i.e. book name and tag value, define book type. The same book type can be stored in more than one library.

1. A library receives a borrow request for a book and a period of time
2. The period is defined by two dates: the request date and the due date.
3. If the book is part of library inventory and the book is available, then the book could be borrowed.
4. If the book is not part of library inventory or the book is already borrowed, the request is sent to another library. If the request cannot be satisfied, then it is dropped.
5. There are no data structures such as queues for keeping track of the requests. However, the inventory system works based on the first-come, first-served principle.

A template of the required classes is given to you. Some methods are complete, some are partially implemented and some have only the signature defined. You can find the project template on your online course web page (scroll down to the last item). It is the last course item called *Assessments*. On the assessment section scroll down for *Assignment 2* link.

For example, the class `Book` has been defined as:

```
class Book {

String         bookName;    // the book name
int            valueTag;    // an integer between -100 and 100
Library        library;     // the library having this book it its inventory
RentSettings   rs;          // rent settings
 ...

  // inner class
  private class RentSettings {
       private String  rentDate;            // date when the book is requested
       private String  dueDate;             // date when the book must be returned
       private boolean borrowed = false;    // true if the book is borrowed
    ...
  }
}
```

The most important method of `Book` class is
`public boolean rentBook(String rentDate, String dueDate, Library library) { ... }`

The method takes three arguments: two strings dates for the renting period and a library from where the book is borrowed. If dates are not valid it creates `DateFormatException` and returns false. If `rentDate` > `dueDate` throws `RentPeriodException` and returns false.
If no exceptions occur a `RentSettings` object must be created for the book that is borrowed.

The `DateFormatException` and `RentPeriodException` are user-defined exceptions you have to create and they are part of project template.

The class `Library` has two fields: a library name and a collection of books kept by the library (data structure `Vector`).

```
public class Library implements MaxTagValue {

    String       libraryName;  // library name
    Vector<Book> books;        // data structure that keeps all books

       ...
}

public interface MaxTagValue {

    /**
     * The method returns an integer.
     * The integer is the greatest value of all tagValues of the library books
     */
    int findMaximumValueTag();
}
```

The most important method of `Library` class is
`public boolean rentRequest(Book wanted, String requestDate, String dueDate)`

The method takes as arguments:
-The book that needs to be rented
-The date when the book can be rented
-The date when the book must be returned to the library
The method returns true, if the book can be borrowed from this library, and false otherwise

The class `Libraries` creates all the `Library` objects and stockpiles the books in each library.

```
public class Libraries {

    public Library[] libraries;        // an array that stores all libraries
    public int numberOfLibraries;      // number of libraries in collection
       ...
}
```

The books definitions are read from a file with the following structure: *book name, tag value*.
For example a file named `NewnhamLibrary.txt` contains lines that define few *books*

```
Database Management Systems,53
Fundamentals of Database Systems (7th Edition),-55
Database System Concepts,45
A First Course in Database Systems (3rd Edition),-20
Essentials of Database Management,25
Database Systems Using Oracle (2nd Edition),65
Database System Concepts,20
Essentials of Database Management,65
```

A library object can be built from a file with the method.
`public Library buildLibraryFromFile(String libraryName, String fileName)`

An important method is the one that takes as arguments the book that needs to be rented with the dates. It searches all libraries for a book that can be rented. It returns the `Library` object where the book is available, if any. If all books are rented returns `null.`

There are other auxiliary classes that will help you to build the required functionality. For example the class `Helper` contains static methods for date validation.

```
public class Helper {

    public static boolean isValidDate(String date) {
        boolean valid = true;

        DateFormat formatter = new SimpleDateFormat("MM/dd/yyyy");
        formatter.setLenient(false);
        try {
            formatter.parse(date);
        } catch (ParseException e) {
            valid = false;
        }
        return valid;
    }
```

The classes of your project must implement the methods: `toString(), equals(), hashCode().` All classes and methods must have *javadoc documentation*. The project framework is defined on codeboard.io.

In order to build and run a tailored personal project your solution must have a number of libraries equal to your last digit of your student ID. If this number is less than 2, than you have to have at least 2 libraries. For instance, if your student ID is *354874345* than your project must create 5 libraries (two libraries are given to you `Seneca@York` and `Newnham,` so you have to create additional 3 libraries).
Each library you create must contain a number of books equal to the second last digit of your student ID. If this number is less than 4, you have to have at least 4 books in each library. You can work with as many books as you like.

Several implementation clarifications:
1. You can assume that files are properly formatted and any line contains *book name, tag value*.
   You do not have to validate. However you can skip the lines that are no correctly structured.
2. The book is defined by a name and a tag value. The tag that is unique for a book with random values between -100 and 100.
   You can assume that a book can appear more than once in a library.
3. When you develop your solution be sure you DO NOT CHANGE the signatures of the given methods. All method in the template must be implemented. However, you can add new methods to enforce the behaviour defined by the project requirements.
4. You must use print methods of the Helper class, when you need to display the rented status of a book.

To demonstrate your solution you have to write code to implement the following requests:

TASK 1
- build libraries from files - at least two libraries
- print library and the books from it

The book must be printed with the following format:
(book name, value tag) - if the book is not assigned to a library, and
(book name, value tag => library name) RentSettings (rent date, due date, library name, borrowed) - if the book pertains to a library.

TASK 2
- ask for a book that does not belong to any library inventory.

TASK 3 - ask for a book that is in a library inventory
- issue a borrow request and print the book object

- issue the same borrow request and print the book object
- return the book
- issue the borrow request with new dates and print the book object

TASK 4 - ask for the same book in all libraries
- look for the same book in all libraries and return a library where the book is in the library inventory
(the library returned is the first library found from the array of libraries)
- look for the same book in all libraries and return a library where the book is available to be borrowed.

TASK 5
- calculate the greatest value tag of all the books from a library

TASK 6
- inquire about a book: is it borrowed?, is it overdue?, could it be found in more than one library?, when can it be borrowed?

3. Submission Requirements – two mandatory procedures

Procedure 1.
Zip only the Java files. The zip file should be named after your last name followed by the first 3 digits of your student ID. For example, if your last name is *Savage* and your ID is *354874345* then the file should be called *Savage354.zip* Upload the zip file to Blackboard

Procedure 2.
Cut and paste your code with documentation on your course page on codeboard.io project called *Assignment 2*. Java Change the code for reading the file from *codeboard.io* (it is given to you).
The code must completely compile and run on *codeboard.io*

4. Marking Scheme

Every class must be properly documented using *javadoc* style
The core classes must implement `toString, equals,` and `hashCode`
Follow strictly Java code conventions.

Note: tasks from 1 to 3 will be marked as assignment 2, and tasks from 4 to 6 will be marked as assignment 3.
However, since the submission date and content for both assignments are the same you will receive a single mark out of 20 marks, instead of one out of 10 from assignment 2 and one out of 10 from assignment 3.

5. Bonus

Each one of these tasks will be worth extra marks:

TASK 7:
- If a book is rented from all libraries, find a library that has this book available closest to the requested date.

TASK 8:
- Instead of using `Vector<Book>` in class `Library` a class `LinkedList<T>` and instantiated as `LinkedList<Book>`

Here is an example of a potential output (your output could look different, but must implement the given tasks):
(if you work with files such: YorkLibrary.txt – you can assume that the book title does not contain comma)

Java Programming Languages,53
A First Course in Database Systems,25
Introductory Statistics with R,-95
Java: The Complete Reference Ninth Edition,45

```
* TASK 1 *
Library = Seneca@York
[
(Java Programming Languages, 53 => Seneca@York)
(A First Course in Database Systems, 25 => Seneca@York)
(Introductory Statistics with R, -95 => Seneca@York)
(Java: The Complete Reference Ninth Edition, 45 => Seneca@York)
]
Library = Newnham
[
(Database Management Systems, 53 => Newnham)
...
]

 * TASK 2 *
Book (C++, 20) does not exist!

 * TASK 3 *

 * TASK 4 *

 * TASK 5 *

 * TASK 6 *
```