Programming Assignment #4

Due: 11:59pm, Wednesday, March 15
Deadline: 11:59pm, Saturday, March 18

Automated Library

You are hired to develop a library system for the main library at University of West California. Once your system is deployed, the library will be operated mainly by computers, except for a few necessary librarians to help users, shelve books, and maintain the online catalog and user information. For example, "check-out" will be automated (similar to the UCSB library). Your system will maintain and manage information of the books and users for the library. It must be able to

- maintain a catalog for books in the library,
- maintain information about users (students and university employees) including librarians who manage the online information,

∗ provide access for each library user to his/her own personal data and loaned materials from the library, and

- provide access for librarians to update the users' information.

In this programming assignment, you are asked to develop a component of the system to support some of the user access functions (the item marked "∗" in the above list) described in the remainder of this document. Note that some of the functions will be adapted to the assignment setting, e.g., email messages or SMS texts could be simply displayed in the console output instead of being actually sent.

Details

Users Library users are *students*, *graduate researchers*, and *professors*. Generally, each user has a name (a string of length up to 30 characters including spaces), an email address (a string), a phone number (10 to 15 digits), a unique library ID (an integer), and a PIN (*Personal Identification Number*, an integer). In addition, associated with each user is a list of books checked out by the user.

The Catalog The library maintains a collection of books. Each book item has a unique "magic" number, and the following attributes: the name(s) of its author(s), the publisher, the title, the year of publication, and a subject.

A *magic* number consists of three parts separated by "." (dot):

1. The first part has one or two letters (for the main classification).
2. The second part has up to 4 characters (alphanumeric) used for subclassification.
3. The third part is up to 10 digits as the serial number within the subclassification.

Note that while magic numbers are unique for book descriptions, there may be multiple (physical) copies with identical magic numbers. Thus, in addition to a magic number, there is a *copy* number (up to 2 digits) for each book (i.e., copy 1, 2, etc.). Each physical book has a unique magic number copy number combination.

Loan Policies When a student checks out an item, the item must be returned within 20 days (e.g., if it is checked out on March 15 and it must be returned on or before April 4). [1] A graduate researcher can check out an item for up to 30 days. And a professor can check out an item for up to 40 days.

[1] You can assume that February has 28 days.

In addition, there are limits on the number of books on loan to a borrower at the same time. The limits are: 10 for a student, 15 for a graduate researcher, and 20 for a professor.

Note that both these limits and loan lengths are subject to change. Policy change operations are not included in this assignment, but your design should be allow these changes be done easily, i.e., the limits should *not* be hard coded.

If the user has at least one overdue items, the user is *blocked*. No blocked users can check out any items.

Operations Your system supports a set of operations and requires the user to log in with a valid a library ID and PIN (clear PIN is fine, no need to encrypt) before any of these operations. In order to avoid entering ID/PIN repeatedly, you can group multiple operations into a *session*. The user enters a session with valid ID/PIN, performs needed operations, and terminates the session explicitly.

Your system must support the following operations for the users. You will decide on the actual interface for your system, standard text input/output would be just fine.

The user should be able to perform the following operations for searching/checking in/out materials. (For the ease of referencing these operations, we use a single letter to represent the operations.)

- (S) *Search* for books by specifying a *keyword*. If a book description contains the *keyword* in the author names, the title, subject, or publisher, or matches the publishing year or the magic number, the book will be displayed in the result. Searching is case insensitive. The search result should contain book descriptions, i.e., multiple copies of the same book has a single entry in the result.
- (D) *Display book information*. Given a magic number, this operations lists all copies of the book and their loan status, i.e., available, on loan till March 10, etc.
- (C) *Check out* a book. Given a magic number and a copy number, the operation will record in the system that the copy of the book is checked out. If the user is blocked or the book is not available, the operation fails with no change in the system (data).
- (R) *recall* and (W) *wait for* a book. The operations need a magic number as the input. If there is a copy of the book available, both operations simply print a message informing the user of availability. When no copies of the book are available, the recall operation sends an email to each user who currently holds a copy of the book requesting the return of the book in 3 days, regardless of the original due dates. Note that the new due dates should also be recorded in the system. the wait-for operation simply puts in the system a reminder such that when a copy of the book is returned, the user gets an SMS text notification of availability. The user will continue to get such SMS texts until the user checks out a copy of this book.
- (T) *Return* a book. It simply reads the magic number and copy number from the reader, and then updates the relevant data.
- (U) *Show* all user information including blocking status and all books checked out by the user with their due dates.

Note that for each of the operations your interface should have an entry to manually enter the date. Alternatively, the date could be entered when a session starts.

An input file named `Library.data` will be provided for your test runs. The input file contains information about books, and about users including their checked-out items. Your application should load this input file at the beginning, and provide operations based on this data set.

The file `Library.data` has four parts. the first part starts with

```
::::::::::
Books
::::::::::
```

and contains 7 lines for each book, one for each of magic number, title, author(s), publisher, year, subject, and copy number (in this order). The second, third, and fourth parts contain user information that start with similar headings:

```
::::::::::          ::::::::::               ::::::::::
Students     or     Graduate Researchers  or  Professors
::::::::::          ::::::::::               ::::::::::
```

Each student/graduate researcher/professor contains the follwing information (in this order): library ID, PIN, name, email address, phone, and a sequence of pairs of check-out date and magic-copy number pairs (books checked-out) if any (otherwise this line is blank).

You can examine the file `Library.data` to see the details.

What to Turn in

You should turn in the complete set of your Java classes for this assignment, including the one with the `main` method. Name the class with the `main` method `LibrarySystem`. Your classes should be in the package `ProgAssignment4`.

You are required to turn in the following electronically:

1. A UML class diagram for all classes in your program. (If you draw by hand, take a picture and turn them into a pdf file.) Name this file `uml.pdf`.
2. A list of design patterns used in your program. For each design pattern, provide the class diagram for the pattern. Name this file `patterns.pdf`.
3. All Java source code files (i.e., everything`.java`). At least `LibrarySystem.java` should have your name, perm number, and email address at the beginning.
4. You should turn in one of the following:

   A *short video* (`demo.`*zzz*) showing execution of the *sample test operations* listed blow. (Here *zzz* is the extension for the video format you use.) Your video should last at most 3 minutes.

   Or, alternatively, a sequence of pictures or screenshots indicated in the *sample test operations*.

Sample test operations (`pic`i means you should take a picture/screenshot and name the file `PIC`i`.png`, `gif` and `jpg` are also acceptable; if you have several pictures to show for one step, name them `pic`i-j):

   (a) Launch your application with the user interface. `pic01`
   (b) Log in with library ID/PIN: `2345678` / `2345` and the date `2017/3/10`. `pic02`
   (c) Select search (S) operation and enter the keyword "`database`". `pic03`
   (d) Perform the S operation. `pic04`
   (e) Display (D) book information for `QA.76.957312`, copy `1`. `pic05`
   (f) Check out (C) this book. `pic06`
   (g) Recall (R) book `QA.76.933653`. `pic07`
   (h) Wait for (W) book `QA.76.943744`. `pic07`
   (i) Show (U) user information. `pic08`
   (j) End the session, logout. `pic09`
   (k) Log in with library ID/PIN: `3456789` / `3456` and the date `2017/3/12`. `pic10`
   (l) Return (T) the book `QA.76.943744`, copy `1`. `pic11`
   (m) Return (T) the book `QA.76.933653`, copy `1`. `pic12`
   (n) log out and end the session.

5. A file named `README.txt` that explains how to start and run your application.

You can turn in all files using the TURNIN command:

```
turnin assign4@cs56 LibrarySystem.java file ··· file
```

at the UNIX prompt.