University library management system

Part one (worth 75% of the grade)

You are required to develop a simple library management system for some university using the Java language and following the concepts of Object-oriented programming you have studied so far. There are only two types of users in this system: Admins, Librarians.

The admin is responsible for adding new librarians to the system and removing the data of the librarians that no longer work for the university.

In order to model the admin and his responsibilities in our system, we need to define a few classes.

First Class:

We need to define a class named LibrarianUser to represent a librarian's personal information. There are five private String variables in the class:

- librarianId
- Name
- Email
- Address
- PhoneNumber

There is only one constructor in the class

- public LibrarianUser(String librarianId, String name, String email, String address, String phoneNumber)

The class contains two methods:
- public String lineRepresentation(): returns the data of the librarian comma separated.

- public String getSearchKey(): returns the librarian id.

The data of the librarians is stored in a file named Librarians.txt. Each line represents a data record of one librarian. For example, the following line represents a librarian whose librarian id is L1200, name is Ahmed, email is ahmed_1991@gmail.com, address is Alexandria and phone number is 01088877345.

L1200,Ahmed,ahmed_1999@gmail.com,Alexandria,01088877345

Note: The librarian id is unique for each librarian.

Second Class

Define a class named LibrarianUserDatabase which is responsible for accessing the file of the librarians (reading from and writing to), accessing and manipulating (adding to and removing from) the list of LibrarianUser objects that hold the data read from the file.

There are two private variables in the class:

- records (ArrayList<LibrarianUser>)
- filename(String).

There is only one constructor in the class:

- public LibrarianUserDatabase(String filename)

The class contains eight methods:

1. **public void readFromFile()**: opens the file whose name is stored in the instance variable filename, reads all the records of the librarians and stores them in the arrayList referenced by the instance variable records.
2. **public LibrarianUser createRecordFrom(String line)**: returns a LibrarianUser object whose arguments are stored in the parameter that named line comma separated.

3. **public ArrayList<LibrarianUser> returnAllRecords()**: returns a reference on the arrayList referenced by the instance variable records.
4. **public boolean contains(String key)**: searches the arrayList referenced by the instance variable records and returns true if there is a LibrarianUser object whose librarian id equals the parameter key.
5. **public LibrarianUser getRecord(String key)**: searches the arrayList referenced by the instance variable records and returns a reference on the LibrarianUser object whose librarian id equals the parameter key.
6. **public void insertRecord(LibrarianUser record)**: inserts the object referenced by the parameter record into the arrayList referenced by the instance variable records .
7. **public void deleteRecord(String key)**: searches the arrayList referenced by the instance variable records and deletes the LibrarianUser object whose librarian id equals the parameter key.
8. **public void saveToFile()**: deletes the old data stored in the file whose name is stored in filename, writes the data stored in the arrayList referenced by the instance variable records to the file. Each line represents the data of one librarian comma separated.

Third Class

Define a class named AdminRole that represents the role of the admin. The class contains only one private variable named **database** which is an object from the LibrarianUserDatabase class.

There is only one constructor in the class and its header is **public AdminRole()**

The class contains four methods:

1. **public void addLibrarian(String librarianId, String name, String email, String address, String phoneNumber):** adds a new librarian to the file named Librarians.txt
2. **public LibrarianUser[] getListOfLibrarians()**: returns an array that contains all the librarians stored in the file named Librarians.txt
3. **public void removeLibrarian(String key)**: removes the librarian whose librarian id equals the parameter key from the file named Librarians.txt
4. **public void logout()**: writes all unsaved data to the file named Librarians.txt

Now, we will describe the role of the librarian and the classes needed for this role.

The librarian is responsible for adding new books to the library, issuing books to students, receiving the books, and adding them back to the library when the students return them. So we need some more classes for applying the librarian's role.

First Class

Define a class named Book to represent a book's information.

The class contains four private String variables:

- bookId
- title
- authorName
- publisherName

One private int variable

- quantity.

The quantity variable represents the number of copies of a book available to be borrowed. When a student borrows a copy of the book, the quantity decreases by one and when he returns it, the quantity increases by one. If quantity equals zero, it means that all of the copies have been borrowed but have not yet been returned.

There is only one constructor in the class and its header is

public Book(String id, String title, String authorName, String publisherName, int quantity)

The class contains four methods:

1. **public int getQuantity()**
2. **public void setQuantity()**
3. **public String lineRepresentation():** returns the data of the book comma separated.
4. **public String getSearchKey():** returns the book id.

The books' data are stored in a file named **Books.txt**. Each line represents a data record of one book. For example, the following line represents a book whose book id is B2394, title is Physics, author name is Albert Einstein, publisher name is Princeton University, and quantity is 10.

B2394,Physics,Albert Einstein,Princeton University,10

Note: The book id is unique for each book.

Second Class

A class named BookDatabase is responsible for accessing the file of the books (reading from and writing to), accessing and manipulating (adding to and removing from) the list of Book objects that hold the data read from the file.

There are two private variables in the class:

- records (ArrayList<Book>)
- filename (String).

There is only one constructor in the class and its header is

public BookDatabase (String filename)

The class contains eight methods:

1. **public void readFromFile():** opens the file whose name is stored in the instance variable **filename,** reads all the records of the books and stores them in the arrayList referenced by the instance variable **records.**
2. **public Book createRecordFrom(String line):** returns a Book object whose arguments are stored in the parameter line comma separated.
3. **public ArrayList<Book> returnAllRecords():** returns a reference on the arrayList referenced by the instance variable **records.**
4. **public boolean contains(String key):** searches the arrayList referenced by the instance variable **records** and returns true if there is a Book object whose book id equals the parameter key.
5. **public Book getRecord(String key)**: searches the arrayList referenced by the instance variable **records** and returns a reference on the Book object whose book id equals the parameter key.
6. **public void insertRecord(Book record)**: inserts the object referenced by the parameter record into the arrayList referenced by the instance variable **records** .
7. **public void deleteRecord(String key)**: searches the arrayList referenced by the instance variable **records** and delete the Book object whose book id equals the parameter key.
8. **public void saveToFile():** deletes the old data stored in the file whose name is stored in **filename**, writes the data stored in the arrayList referenced by the instance variable **records** to the file. Each line represents the data of one book comma separated.

Third Class

Define a class named StudentBook representing the borrowing relationship.

The class contains two private String variables:

- studentId
- bookId

One private LocalDate variable

- borrowDate.

The studentId variable represents the id of the student who has borrowed a copy of the book whose id is stored in the bookId since the date stored in borrowDate.

Each object of the StudentBook class represents an unfinished borrowing operation where the student hasn't returned the book yet.

There is only one constructor in the class and its header is

public StudentBook(String studentId, String bookId, LocalDate borrowDate)

The class contains five methods:

1. **public String getStudentId()**
2. **public String getBookId()**
3. **public LocalDate getBorrowDate()**
4. **public String lineRepresentation():** returns the data of the object comma separated.
5. **public String getSearchKey()**: returns a String value = studentId +","+bookId

The data of unfinished borrowing operations is stored in a file named StudentsBooks.txt. Each line is a data record for a single borrowing operation. The following line, for example, represents an object whose student id (The borrower) is 7845, book id that has been borrowed is B2568, and borrow date is 12-02-2022.

7845,B2568, 12-02-2022

<u>Note</u>: If a student borrows a copy of a book and does not return it, he cannot borrow another copy of the same book until he returns the one he has. As a result, the combination of the student id and the book id is distinct for each borrowing operation.

<u>Fourth Class</u>

A class named StudentBookDatabase is responsible for accessing the file of the unfinished borrowing operations (reading from and writing to), accessing and manipulating (adding to and removing from) the list of StudentBook objects that hold the data read from the file.

The class contains two private variables:

- records (ArrayList<StudentBook>)
- filename(String).

There is only one constructor in the class and its header is

public StudentBookDatabase (String filename)

<u>Note</u>: It should be noted that the argument key supplied to the methods described below is a String made up of two parts separated by a comma. The first part is a student id, while the second part is a book id.

The class contains eight methods:

- **public void readFromFile():** opens the file whose name is stored in the instance variable **filename,** reads all the records of the books and stores them in the arrayList referenced by the instance variable **records.**
- **public StudentBook createRecordFrom(String line):** returns a StudentBook object whose arguments are stored in the parameter line comma separated.
- **public ArrayList<StudentBook> returnAllRecords():** returns a reference on the arrayList referenced by the instance variable **records.**
- **public boolean contains(String key):** searches the arrayList referred to by the instance variable **records** and returns true if there is a StudentBook object whose studentId and bookId can be combined to form a String = studentId+","+bookId that matches the parameter key
- **public StudentBook getRecord(String key)**: searches the arrayList referred to by the instance variable **records** and returns a reference to the StudentBook object whose studentId and bookId can be combined to form a String = studentId+","+bookId that matches the parameter key.
- **public void insertRecord(StudentBook record)**: inserts the object referenced by the parameter record into the arrayList referenced by the instance variable **records**.
- **public void deleteRecord(String key)**:searches the arrayList referenced by the instance variable **records** and delete the StudentBook object whose studentId and bookId can be combined to form a String = studentId+","+bookId that equals the parameter key.
- **public void saveToFile():** deletes the old data stored in the file whose name is stored in **filename**, writes the data stored in the arrayList referenced by the instance variable **records** to the file. Each line represents the data of one book comma separated.

<u>Fifth Class:</u>

Define a class named LibrarianRole that represents the role of the librarian. The class contains two private variables named **booksDatabase and sBDatabase** which are objects from BookDatabase class and StudentBookDatabase class respectively.

There is only one constructor in the class and its header is **public LibrarianRole()**

The class contains only six methods:

- **public void addBook(String id, String title, String authorName, String publisherName, int quantity):** adds a new book to the file named Books.txt
- **public Book[] getListOfBooks()**: returns an array that contains all the books stored in the file named Books.txt
- **public StudentBook[] getListOfBorrowingOperations()**: returns an array that contains all the unfinished borrowing operations stored in the file named StudentsBooks.txt
- **public int borrowBook(String studentId, String bookId, LocalDate borrowDate):**
  - returns zero if the quantity variable of the book whose book id matches the parameter bookId is zero.
  - returns one if the student whose student id equals the parameter studentId

    has borrowed a copy of the same book and hasn't returned it yet.
  - If none of the above situations occur, the method decreases the quantity variable of the book by one, adds a new unfinished borrowing operation to the file named StudentsBooks.txt, updates the file Books.txt and returns 2
- **public double returnBook(String studentId, String bookId, LocalDate returnDate):**
  - increases the quantity variable of the book whose book id equals the parameter bookId by one
  - calculates the late return fee
  - removes the line representing the borrowing operation from the file
  - updates the file Books.txt and returns the late fee.

- ○ To calculate the late fee, the method first calculates the difference between the borrow date and the return date in days. Then, it returns a zero fee if the difference is less than 7 days, otherwise it returns (the difference – 7) * 0.5.

- **public void logout()**: writes all unsaved data to the files named Books.txt and StudentsBooks.txt

Part 2 (worth 25% of the grade)

By solving question 1, you have got 75% of the assignment mark. The remaining 25% are assigned for refactoring your code. If you give a look at your code in question 1, you will find many duplicated code. So, your task in question 2 to get the full mark is to enhance your code by applying OOP concepts on your code and make your code cleaner and reusable. OOP concepts like inheritance, polymorphism, abstraction, etc. You must apply OOP concepts on your code to get the full mark.

Note: Refactoring your code means enhancing it. Making your code cleaner and removing the duplicated code is what is meant by refactoring.

Required

1. You are required to solve the first question and get a 75% of the assignment mark. If you want to get the full mark, you should refactor your code and modify it to obey the OOP concepts (inheritance, polymorphism, abstraction, …) as mentioned in question 2.
2. The due time for the required question to be delivered is this Saturday 29 /10 at 10 am.
3. A discussion is made with you at your lab time next week on what you have delivered.

What to be delivered

- On the google form, you should deliver a zipped file that contains the .java files of your classes.
- Your zip file should be named as id_groupNumber. For example, 4678_G2.

Policies:

- You should work in groups of **two**.
- Delivering a copy will be severely penalized for both parties, so delivering nothing is so much better than delivering a copy.
- No late submission is allowed