Advanced Programming Assignment

Library Automation System

Contributors: Swapnil Panigrahi

What is the Project?

The project is a library automation system, which can be accessed by the librarian and the member alike. The librarian has the freedom to add members, remove them, add books, and remove them. Additionally, he can view the member details, such as the books in their possession and current fine as well. The students have the ability to issue books, return them, and pay their fines.

1 Classes and Methods

1.1 `Book`

- **Attributes:**
  1. `ID: int` Unique Book ID
  2. `Title: String` Title of the book
  3. `Author: String` Author of the book
  4. `Copies: int` Total copies of the book in the library
  5. `Issue Time: Instant` Time of issue of the book
- **Methods:**
  1. `getID()` Returns ID of the book
  2. `getIssueTime()` Returns issue time of the book
  3. `isIssued()` Returns issue status of the book
  4. `getAuthor()` Returns author of the book
  5. `getTitle()` Returns title of the book
  6. `getCopies()` Returns no. of copies of the book
  7. `issueBook()` Issues the book, and sets the issue time
  8. `returnBook()` Returns the book
  9. `increaseCopies()` Increases the total number of copies

1.2 `Student`

- **Attributes:**
  1. `Name: String` Name of the member
  2. `Age: int` Age of the member
  3. `PhoneNumber: String` Phone number of the member, which also acts as a unique id for login

4. `Fine: int` Pending fine of the user
5. `Lib: Library` Library the member belongs to
6. `BookIssued: Vector` List of the issued books

- **Methods:**
  1. `getName()` Returns name of the member
  2. `getAge()` Returns age of the member
  3. `getID()` Returns the unique id of the user
  4. `getIssued()` Returns list of the issued books of the member
  5. `getFine()` Returns the pending fine of the member
  6. `view_book()` List all books available to issue
  7. `issue_book()` Issues the book in the name of the student
  8. `return_book()` Returns the book in possession of the user and updates the fine
  9. `update_fine()` Updates the fine of the user
  10. `pay_fine` Clears the fine of the user
  11. `login()` Logs in as a member
  12. `viewIssued()` Lists all the books in possession of the member

1.3 Librarian

- **Attributes:**
  1. `Library` Library which it belongs to
- **Methods:**
  1. `add_student()` Asks for name, age, and phone number from the librarian and passes it to the `add_student` method of the `Library`
  2. `remove_student()` Calls `remove_student` method of `Library`
  3. `view_students()` Lists details of all `student`
  4. `add_book()` Asks for the title and author of the book and passes it to `add_book` method of `Library`
  5. `remove_book()` Asks for book id and passes it to the `remove_book` method of `Library`
  6. `viewAllBooks()` Lists all the books in the `Library`

1.4 Library

- **Attributes:**
  1. `StudentList` List of all `Student` registered
  2. `BookList` List of all `Book` in the `Library`
- **Methods:**
  1. `add_book(String title, String author)` Adds book to the `BookList` with the given title and author attributes and assigns a `ID` to it

2. `issue_book(int ID)` Issues `Book` given the `ID`
3. `return_book(Student student, int bookID)` Returns `Book` of the `Student` given `bookID`
4. `remove_book(int bookID)` Removes the `Book` from the `Library` by removing it from the `BookList`
5. `add_student(String name, int age, String phone)` Adds `Student` to the `Library` with the given name, age and phone number
6. `remove_student()` Asks for input of the `Student ID` and removes the member
7. `listAllBooks()` Lists all `Book` with their properties
8. `listAllStudents()` Lists all `Student` with their properties
9. `getStudent(String ID)` Returns the `Student` given the `ID`

2 Relationships

2.1 Composition

The `Library` class is composed of an array of `Book` objects and an array of `Student` objects. For example, the issue status of `Book` object can be updated by the library, and it also manages the number of copies of the `Book` object.

```
public void remove_book(int bookID){
    if (this.BookList.isEmpty()){
        System.out.println("There are no books in the library!");
        return;
    }
    boolean success = false;
    Book temp = null;
    for(Book i : this.BookList){
        if (i.getID()==bookID && i.isIssued()){
            temp = i;
            success = true;
        }
    }
    if (success){
        System.out.println("Removed Book with ID: "+bookID);
        BookList.remove(temp);
    }
    else{
        System.out.println("No Book found with ID: "+bookID);
    }
}
```

Listing 1: Snippet from the Library Class showing it accessing the Book's ID getter method and Issue Status getter method

```java
public void add_book(String title, String author){
    int bookID=BookList.size();
    int copies = 1;
    for (Book i : this.BookList){
        if (Objects.equals(i.getAuthor(), author) && Objects.equals(i.getTitle(), title)){
            i.increaseCopies();
            copies++;
        }
    }
    new_book = new Book(bookID, title, author, copies);
    BookList.add(new_book);
}
```

Listing 2: Snippet from the Library Class showing it setting the copies of the `Book` object

2.2 Association

The `Librarian` class has an association with the `Library` class, representing that they manage books and students from the library. For example, the `Librarian` class can only access the `Library` class properties using only its getter methods.

```java
public Book return_book(Student student, int bookID){
    for (Book i : student.getIssued()){
        if (i.getID()==bookID){
            System.out.println("The book with ID: "+bookID+" has been returned.");
            i.returnBook();
            return i;
        }
    }
    return null;
}
```

Listing 3: Snippet from the `Library` Class showing it accessing the `Book` Class using the getter methods

2.3 Dependancy

```java
public Student getStudent(String ID){
    for (Student i : StudentList){
        if (Objects.equals(i.getID(), ID)){
            return i;
        }
    }
    return null;
}
```

Listing 4: Snippet from the `Library` Class showing it only passing the `String` as parameter without initialising it

3 Basic Operations

- **Adding Books:** The `Librarian` can add books to the `Library` by taking input of the title and the author and passing it to the `Library` object method of adding a new `Book` object.
- **Borrowing and Returning Books:** `Student` can borrow books by selecting available books from the `Library`. The `Book`'s availability status is updated. Students return books, which are then marked as available.
- **Searching for Books:** `Student` can search for books by `ID`. `Librarian` can search for books and provide information to students.

- **Managing Library Users:** `Librarian` can manage `Student` account and tracking issued `Books`. `Librarian` can penalise the `Student` for returning `Book` late. The `Student` can clear their dues by logging into their account.

How to run the project

1. Unzip the folder in any directory of choice or clone the project from the Github Repository
2. Open the terminal at `..\AP-Assignment-1`
3. Run the following commands in order:
   ```
   mvn clean
   mvn compile
   mvn package
   cd target
   java -jar AP-Assignment-1-1.0-SNAPSHOT
   ```
4. To re-run it, you can simply enter `java -jar AP-Assignment-1-1.0-SNAPSHOT` if you are in the `..\AP-Assignment-1\target` directory