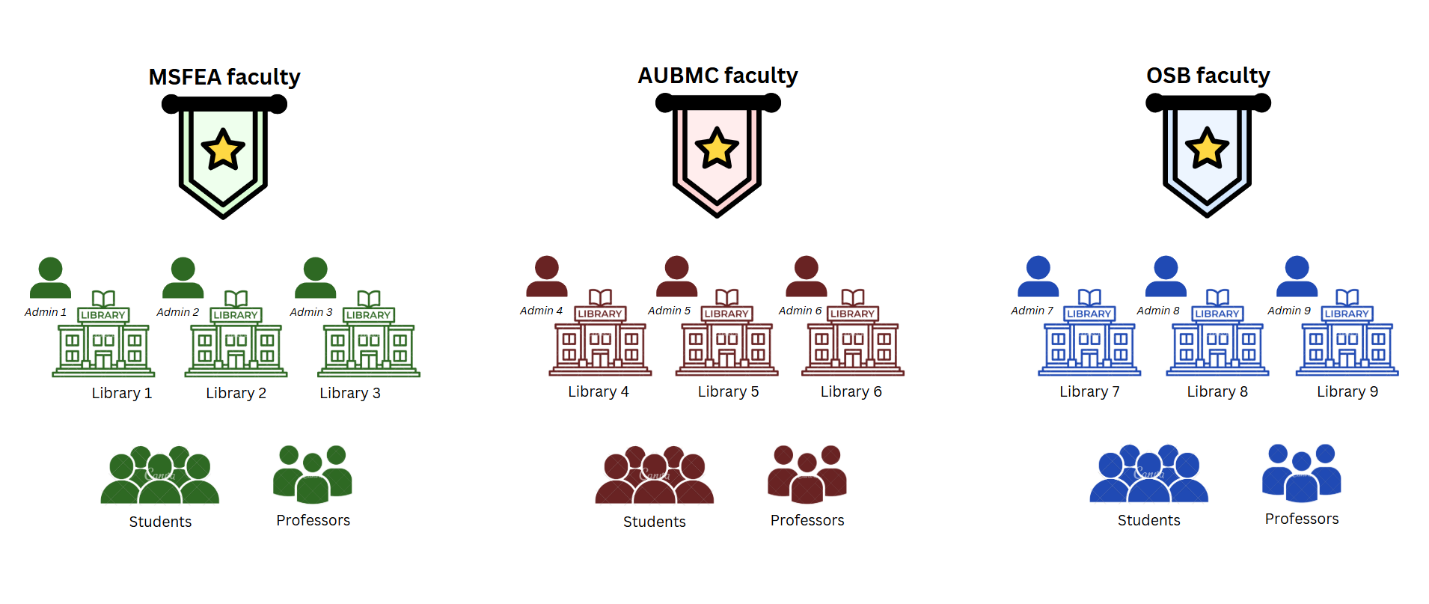
This project aims to develop a library management system that enables each faculty to manage multiple libraries. Each library is assigned to an administrator (librarian) and made accessible to the students and professors within the respective faculty.

***Structured flow diagram of the library management system implemented***

Let’s see how different elements of our library management system are integrated…

1. **Creating users:**

Creating Users: admins, professors and students

Each admin, professor or student is a user in our system; therefore, we create an abstract class called user containing information required by each user of the system. Then, when creating each role, the constructor calls the abstract constructor through super and adds additional attributes unique to each role.

*User input validation:*

When creating a user, it must contain the following:

* Name: the name of the user must only contain alphabetical characters (in our example, we allowed numerical characters because we identify user like student2 or admin8).
* Email: the email must be an AUB email (ends with @mail.aub.edu) and must only contain alphanumeric characters and periods.
* Phone Number: the phone number must only contain numerical characters and be of length 11 (3 for prefix and 8 for the number).
* Address: the address must only contain alphanumeric characters, periods, commas and hyphens.

*Student input validation:*

* By calling super, name, email, phone number and address will be validated
* ID: the ID must only contain numerical characters and be of length 9.

*Professor input validation:*

* By calling super, name, email, phone number and address will be validated
* Rank: the rank can only be “professor” or “lecturer”,

*Admin input validation:*

* By calling super, name, email, phone number and address will be validated
* Schedule: the schedule must only contain alphanumeric characters and hyphens and must be in the format specified (ex: Monday-Friday from 9-3)

1. **Creating Library Resources:**
2. Creating electronic resources: tables and PC

An electronic resource can only be a tablet or a PC; therefore, we create an abstract class electronic device that would represent both tablets and PCs with common attributes. Then, when creating each role, the constructor calls the abstract constructor through super and adds additional attributes unique to each device.

*Device input validation:*

When creating a user, it must contain the following:

* Brand: the brand of the device must only contain alphabetical characters.
* Reference Number: the reference number must only contain numerical characters and be of length.

*Tablet input validation:*

* By calling super, brand and reference number will be validated
* Shelf: the shelf must only contain numerical characters.

*PC input validation:*

* By calling super, brand and reference number will be validated
* Lab Room: the lab room must only contain alphanumeric characters.

Each electronic device will then be saved as electronic resources. The electronic resources class extends the electronic device class making the generic attribute T only represent tablets and PC.

1. Creating rooms:

Each library can contain one or more rooms for meetings.

*Meeting room input validation:*

* Room Number: the room number must only contain alphanumerical characters.
* Contact Information: the contact information is a phone number and must only contain numerical characters and be of length 11 (3 for prefix and 8 for the number).

1. Creating Books:

Each library can contain one or more books.

*Books input validation:*

* Title: The title must contain only alphanumerical characters, spaces, or special characters like periods, commas, colons, apostrophes, and hyphens.
* Author: The author's name must contain only alphabetic characters, periods, commas, hyphens, and spaces.
* Genre: The genre must contain only alphabetic characters and spaces.
* Description: The description must contain only alphanumerical characters, spaces, or special characters like periods, commas, and hyphens.
* Format: The format must be either 'Physical' or 'eBooks'.
* ISBN: The ISBN must contain only numerical characters.
* Publication Year: The publication year must consist of exactly four numerical digits.

It’s important to note that when handling any resource, the process must acquire a lock to view and modify the resource to prevent race conditions.

1. **Exception handing for invalid input:**

By implementing the InvalidInputInformation exception, the system can enforce strict validation rules for user input, ensuring data integrity and preventing errors caused by invalid or inconsistent data. It also enhances user experience by providing clear and actionable feedback when input does not meet the required criteria.

1. **Creating Libraries:**

When creating a library, you must assign an admin to it. This admin will be able to control the resources of his assigned library such as adding books, electronic resources, and rooms as well as deleting them.

Each library will contain:

* The name of the faculty the library belongs to.
* A list of the books the library contains.
* A list of the electronic resources the library contains.
* A list of the meeting rooms the library contains.
* The admin user responsible for the library.

1. **Exception handing for unauthorized user:**

By implementing the UnauthorizedUserAction exception, the system can enforce role-based access control, ensuring that only authorized users can perform specific actions. This prevents misuse or unauthorized operations, enhancing security and maintaining the integrity of the system's functionality.

1. **Rules interface for each ressource:**

By implementing the Rules interface, the system enforces a standardized way to define and manage rules across different resources. This interface should be implemented by each resource and can only be set by the admin of the library containing the resources.

**Rules for msfea:**

* Professors’ late penalty: 0 seconds.
* Students’ late penalty: 5 seconds.
* Professors are allowed 2 renewal passes.
* Students are allowed 1 renewal pass.
* Professors’ rental/booking duration: 10 seconds.
* Students’ rental/booking duration: 7 seconds.
* Professors’ renewal time: 10 seconds.
* Students’ renewal time: 5 seconds.

**Rules for osb:**

* Professors’ late penalty: 0 seconds.
* Students’ late penalty: 7 seconds.
* Professors are allowed 2 renewal passes.
* Students are allowed 0 renewal pass.
* Professors’ rental/booking duration: 7 seconds.
* Students’ rental/booking duration: 5 seconds.
* Professors’ renewal time: 5 seconds.
* Students’ renewal time: 0 seconds.

**Rules for aubmc:**

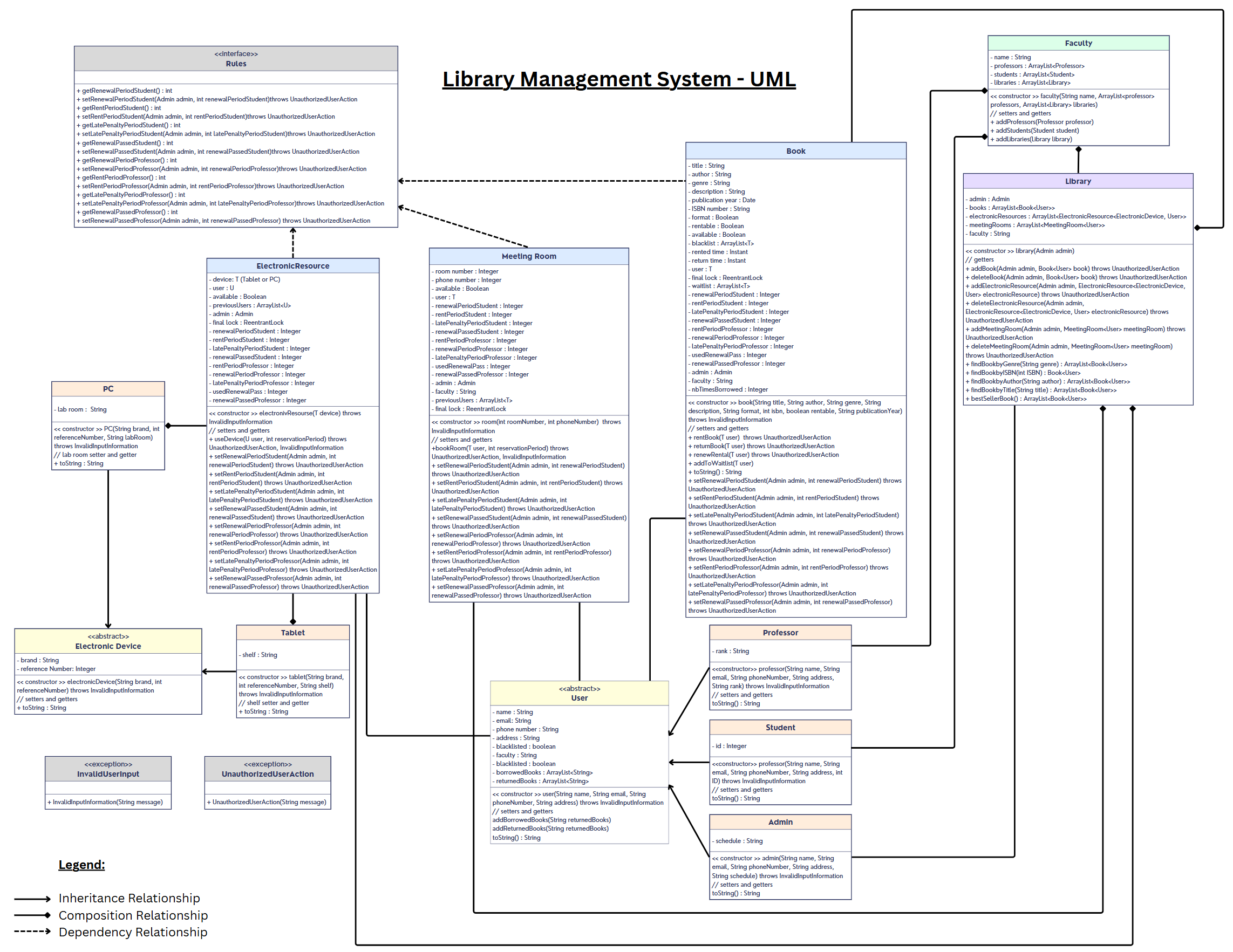
* Professors’ late penalty: 0 seconds.
* Students’ late penalty: 20 seconds.
* Professors are allowed 1 renewal passes.
* Students are allowed 1 renewal pass.
* Professors’ rental/booking duration: 15 seconds.
* Students’ rental/booking duration: 10 seconds.
* Professors’ renewal time: 5 seconds.
* Students’ renewal time: 5 seconds.

1. **Creating Faculties:**

Each faculty created will contain:

* Libraries that belong to the faculty
* Students registered in the faculty
* Professors teaching in the faculty
* The name of the faculty

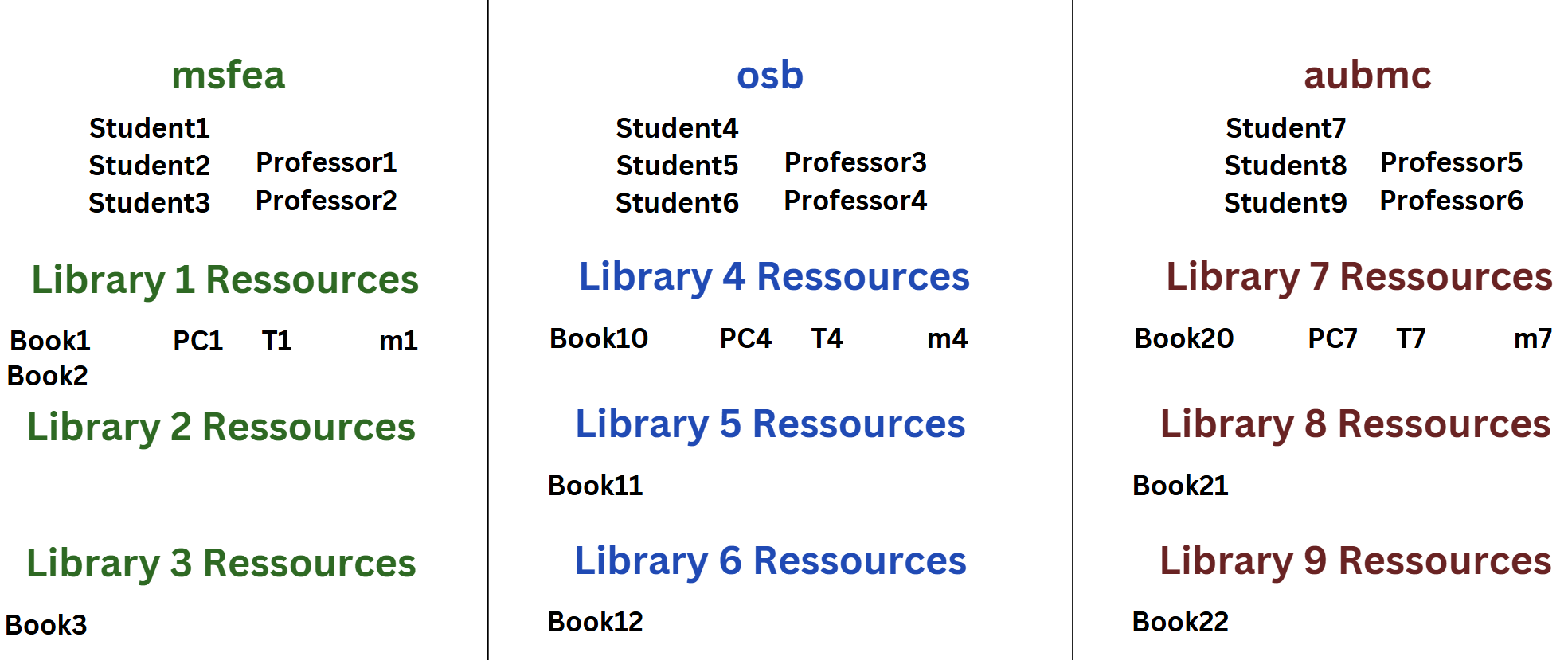
Below is the UML design for our library management system:



Note that:

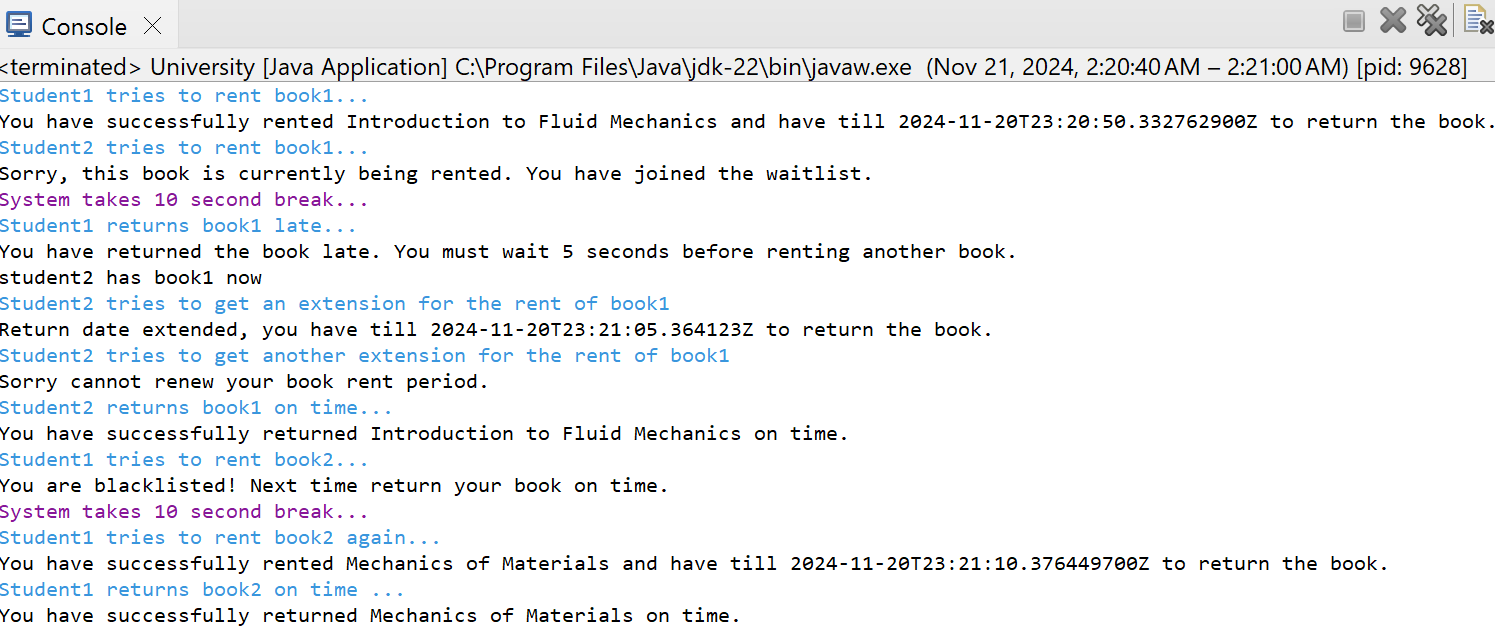
* Each user role (professor, student or admin) inherits from the abstract class user
* Each electronic device (PC or Tablet) inherits from the abstract class electronic device
* Each electronic resource contains either a tablet or PC
* Each meeting room, book, and electronic resource can be associated with a user
* Each meeting room, book, and electronic resource is owned by a library
* Each meeting room, book, and electronic resource implements its own set of rules
* Each library is owned by a faculty
* Each faculty contains students and professors
* Each library is associated with an admin

To simulate a working library management system, I have created three faculties: msfea, osb and aubmc. Then I created 3 libraries for each faculty and assigned an admin for each library and 9 admins, each responsible for one of the libraries created. The admins then proceeded to populate their respective libraries with books, electronic resources and rooms. Then I created students and professors to populate the faculties. Below are the instances created to simulate a live library management system to use as reference for the following scenarios:



We will now simulate different scenarios to showcase the features of our library management system.

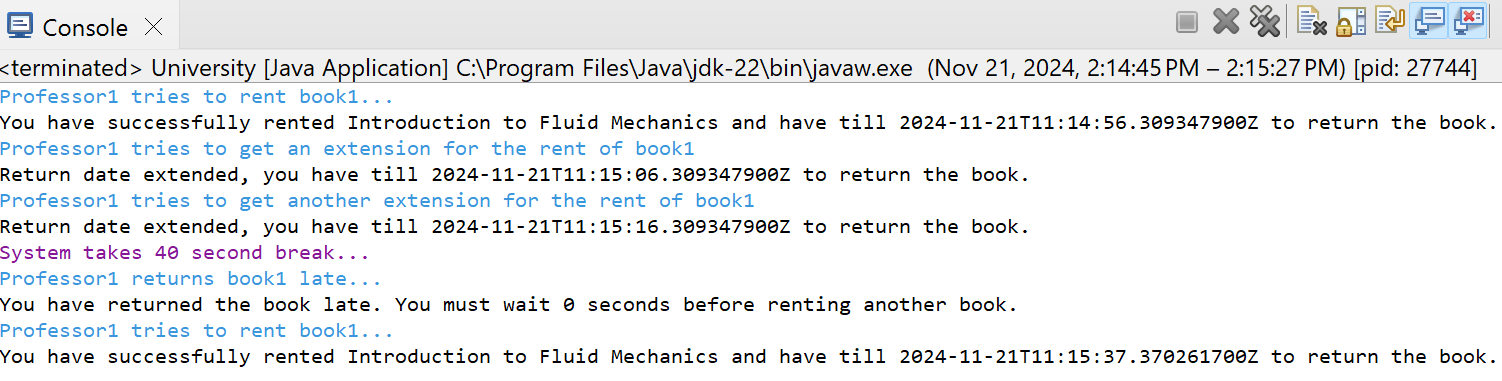
**SCENARIO 1:** Two students from the same faculty are renting books from the faculty’s libraries. (p.s. for the sake of testing, waiting time is in seconds

*Consol output:*

In this scenario applicable for msfea rules, we can see:

* When a book is rented by a student, no other student can rent this book.
* If a student wants to rent a book currently not available, they will be put on the waiting list and once their turn arrives, they will receive the book automatically.
* If a student returns the book late, they are immediately put on a blacklist and must wait a certain amount of time before being able to rent a book again.
* A student can renew their rental only once.

**SCENARIO 2:** A professor tries to rent a book



In this scenario applicable for msfea rules, we can see compared to the one above that:

* A professor has more renewal passes as instated by the admin
* A professor has no late penalty for returning a book late

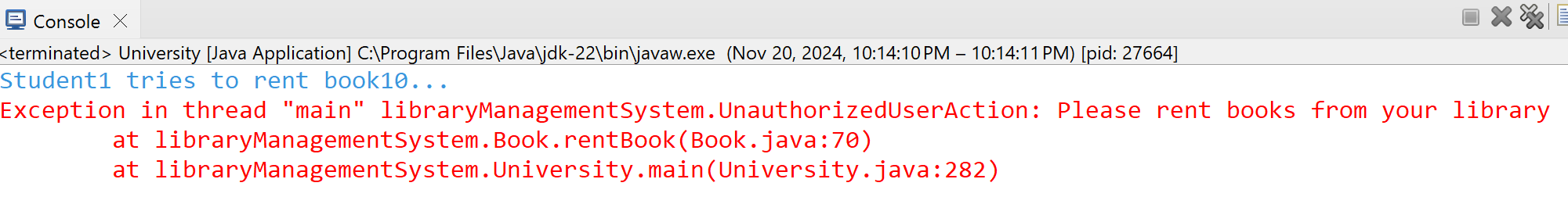
**SCENARIO 3:** A student tries to return a book on behalf of another student.

*Consol output:*A screenshot of a computer

Description automatically generated

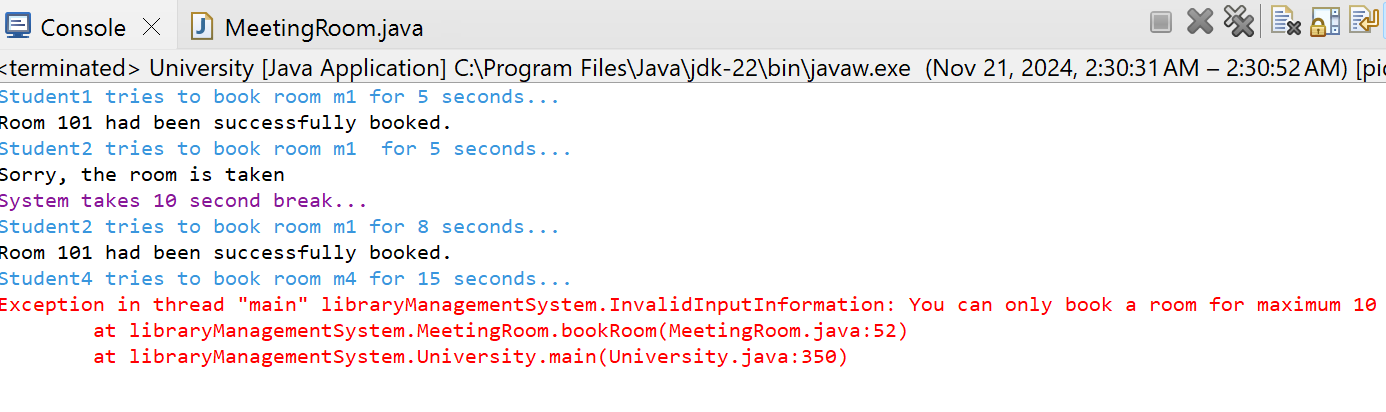
As shown by the output, when a student who doesn’t own the book tries to return it, an exception for unauthorized user action is thrown by the system.

**SCENARIO 4:** A student tries to rent a book from a library outside their faculty

*Consol output:*

As shown by the output, when a student tries to rent a book from outside of their faculty, an exception for unauthorized user action is thrown by the system.

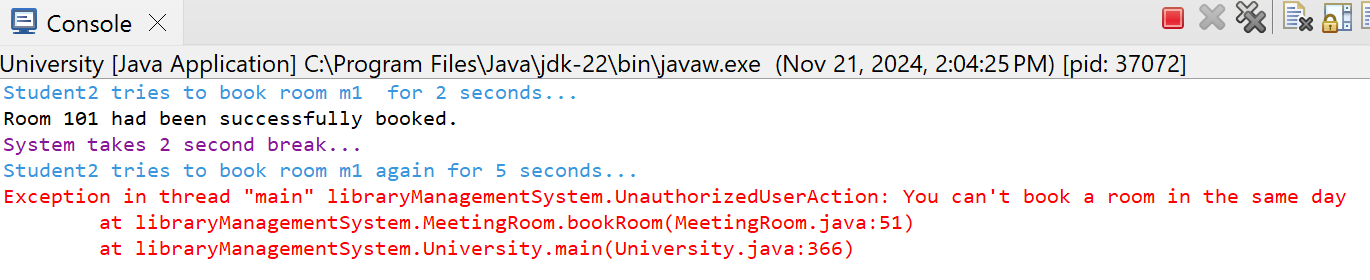
**SCENARIO 5:** Students are trying to rent a room from within their faculty’s libraries

*Consol output:*

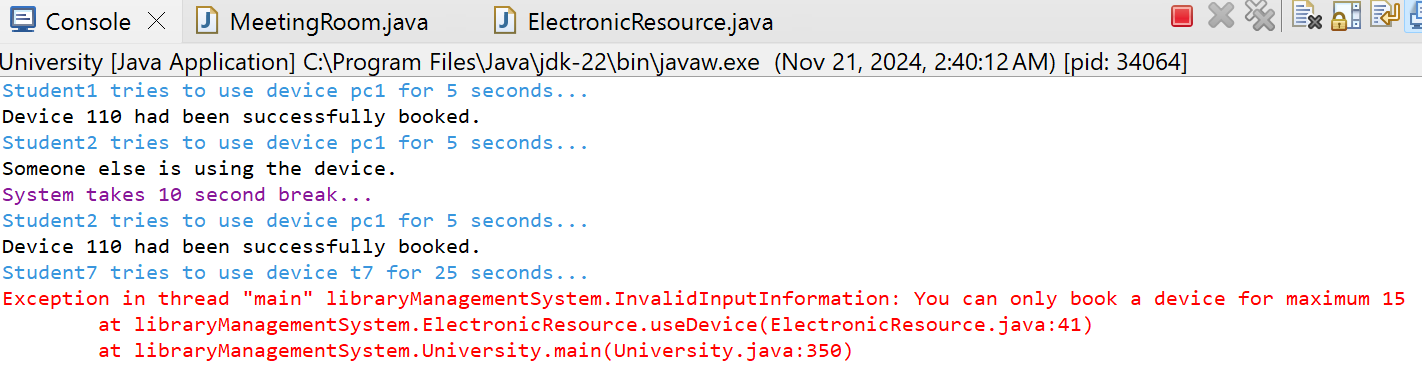
The output shows that:

* If a room is booked by a student, it can’t be booked by any other student
* A student can only book the room for a maximum of 10 seconds as per the osb rules.

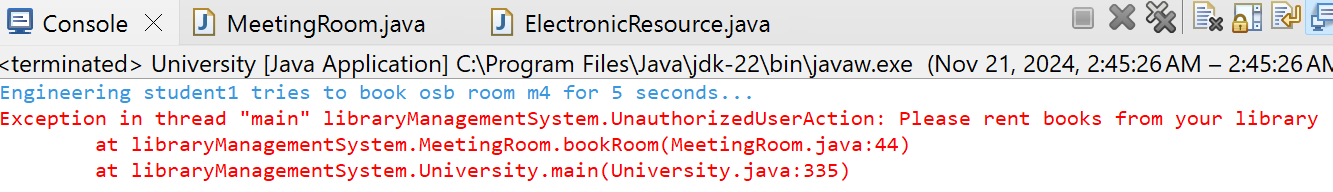
**SCENARIO 6:** Students are trying to rent a room before their renewal time is up



**SCENARIO 7:** Students are trying to use devices from within their faculty’s library

*Consol output:*

**SCENARIO 8:** Students trying to use resources from outside their library

*Consol output:*

*Consol output:*A screenshot of a computer

Description automatically generated

*Consol output:*A screenshot of a computer

Description automatically generated

**SCENARIO 9:** Admins are trying to add resources and set rules to libraries that aren’t theirs

*Consol output:*A screenshot of a computer

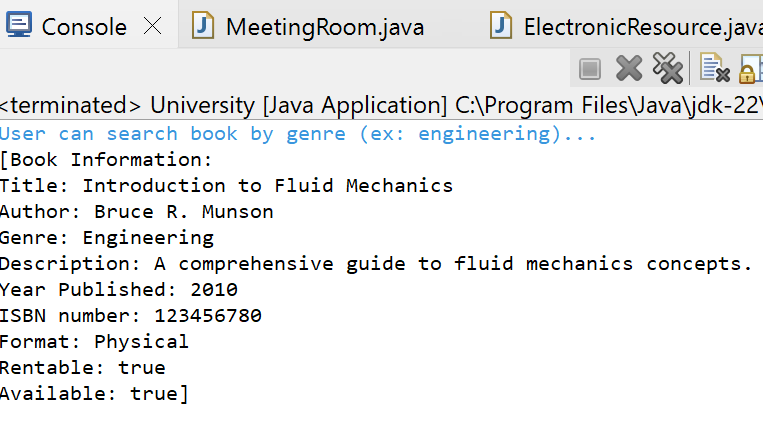
Description automatically generated

*Consol output:*A screenshot of a computer

Description automatically generated

Because the parameters only accept admins, this ensures that professors and students can’t modify the library’s resources.

**SCENARIO 10:** User can find books by genre, but also titles, ISBN, and author

*Consol output:* 

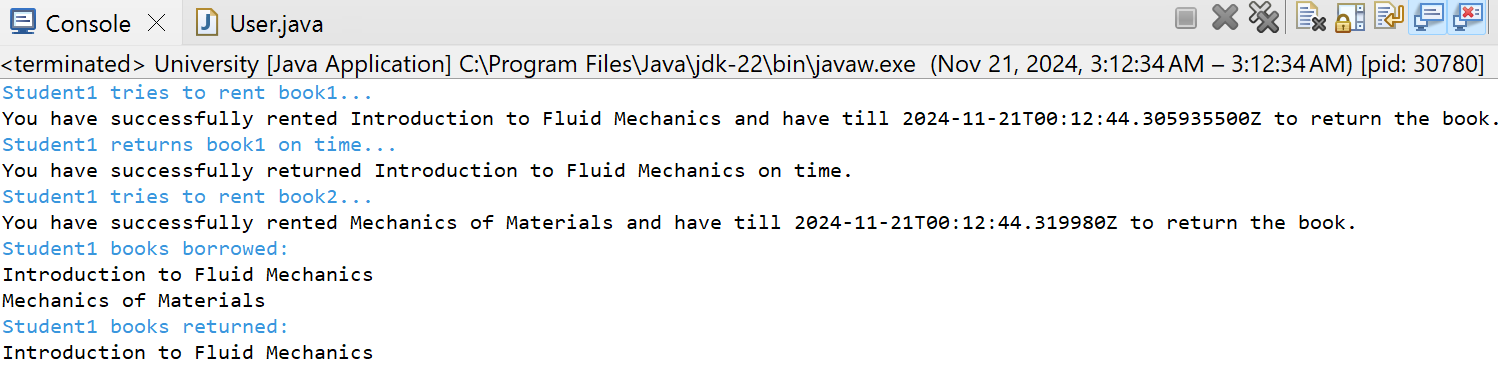
**SCENARIO 11:** We can view all students and professors of a faculty

*Consol output:* A screen shot of a computer

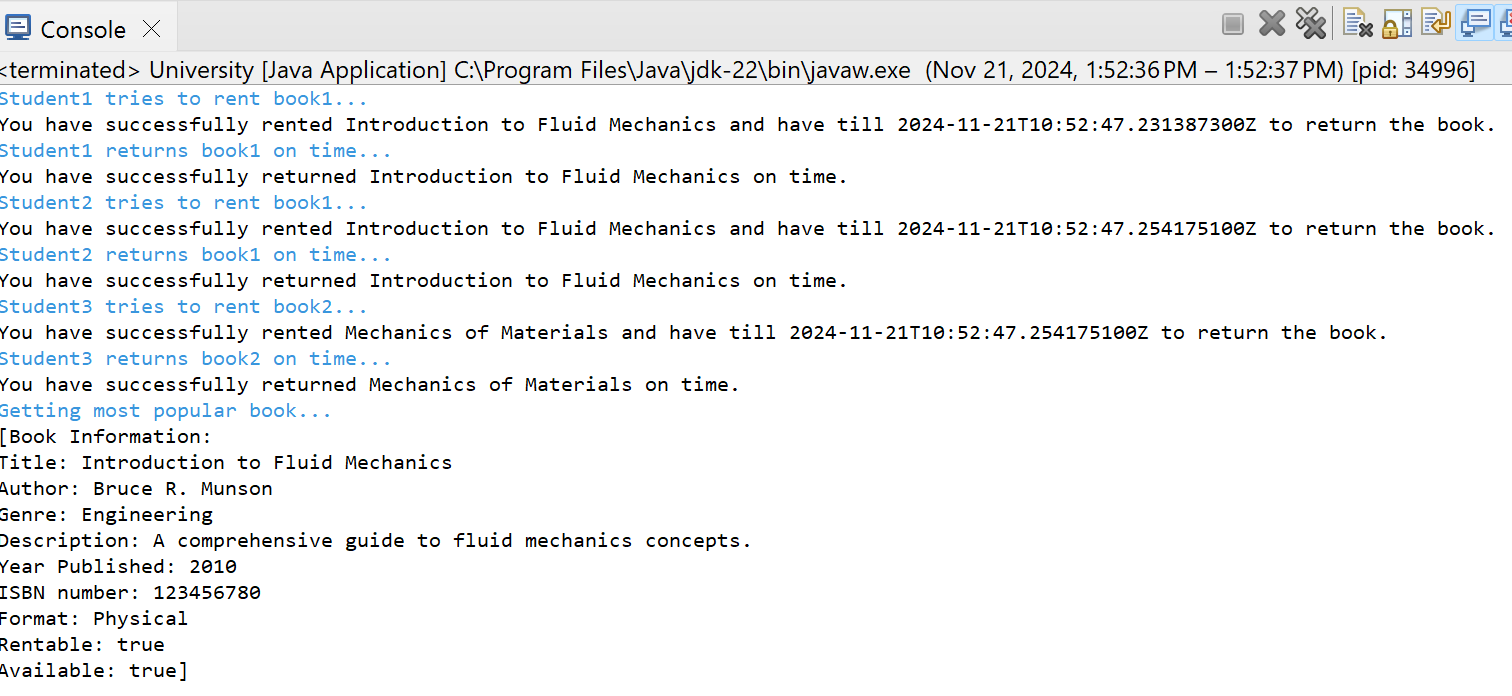
Description automatically generated A screenshot of a computer

Description automatically generated

**SCENARIO 12:** We can view a student’s book history

*Consol output:*

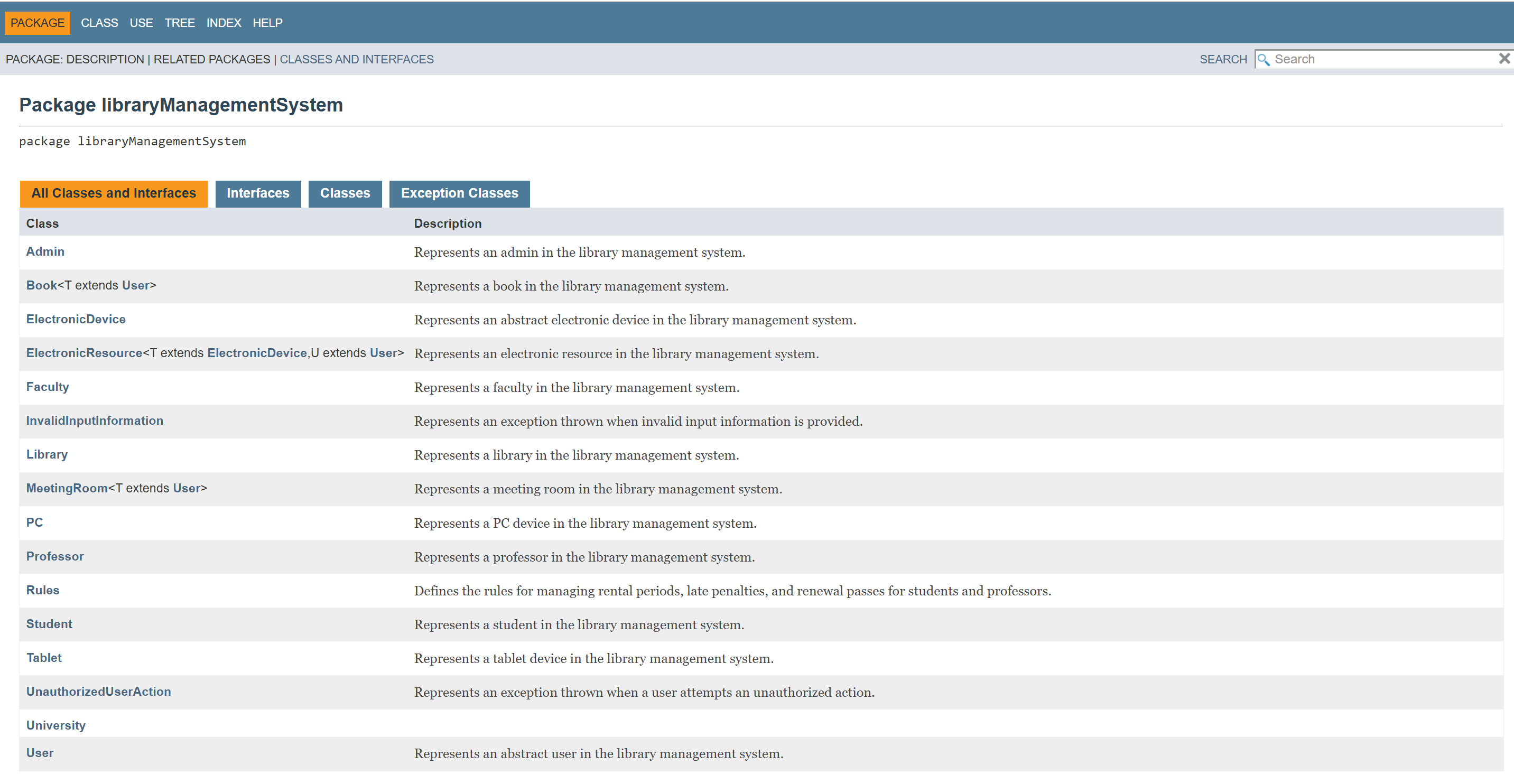
**SCNARIO 13:** Admin can view best seller book



These are the main features of the library management system. Thank you 😊

Extra: Java doc index html

**PACKAGE:**

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**USE:**

A screenshot of a computer

Description automatically generated

**TREE:**

**A screenshot of a computer

Description automatically generated**

**INDEX:**

**A screenshot of a computer

Description automatically generated**