THE UNIVERSITY OF BUEA

****

FACULTY OF ENGINEERING AND TECHNOLOGY (FET)

DEPARTMENT OF COMPUTER ENGINEERING

COURSE TITLE; OBJECT ORIENTED MODELING

COURSE CODE: CEF 331

LIBRARY AUTOMATING SYSTEM

PROJECT REPORT GROUP 4

Name	Matricule
DJEUNOU DJEUNOU MARIEKE JETTIE	FE21A168
DJEUTIO QUOIMON ANDERSON ROY	FE21A169
DJIALEU TANKOUA KEANE RANDY	FE21A170
DJITUE TOGUE BRINDA SPACHELLE	FE21A171
DJOUMESSI DONGMO RONSARD CARNEGIE	FE21A172
DJOUMESSI LEKANE WENDY FORTUNE	FE21A173
DJOUMESSI NGUEFACK IVAN	FE21A174
DUESENBERRY MBIKANG AGBORTAR AKO	FE21A175
EBAI ENOWNKU JANE	FE21A176
EBOMESUMBE NGOLE DANDY BRADLEY	FE21A177
EDI EDISON FORNANG	FE21A178
EFUETANZOH ASONG RODERIC	FE21A179
EKPOMBANG TABENDANG	FE21A180
EKUNDIME GLEAN MAKOGE	FE21A433
ELOMBA KARESLY SAKWE	FE21A181

COURSE LECTURER :DR FOZIN & DR DJOUELA INES

INTRODUCTION

Library management is difficult because librarians have to keep the library organized and stocked. Poorly managed libraries can have many problems such as disorganized shelves, unfilled book orders, outdated reference and more. It is where **Library automating software** helps extensively. It eliminates the need for a human being to do this monotonous task of organizing and updating catalog.

FUNCTIONAL REQUIREMENTS

We have three types of functional requirements which are;

- System requirements
- User requirements
- System Analyst

❖ SYSTEM REQUIREMENTS

- ➢ Keep track of books available in the library.
- ➢ Record name, identification number and password of each customer.
- ➢ Add and acquire new books digitally.
- ➢ Allows members to login, search, select, issue and return books by themselves.
- ➢ Determine when materials are due to be return.
- ➢ Keep track of financial obligations.
- ➢ Send notifications to customers.
- ➢ Provide certain criteria which makes the customer illegible to borrow a book.
- ➢ Store information about the types books highly demanded by customer at a given time.

❖ USER REQUIREMENTS

- Customers should be able to borrow a book.
- Customer should be able to research a book by title or author.
- Customer should be able to pay fines.
- Customer should be able to authenticate.
- Admin or Liberian should be able to check books availability.
- Customer and admin should be able to search or locate a book by the unique ID code, title, and author or publication date.

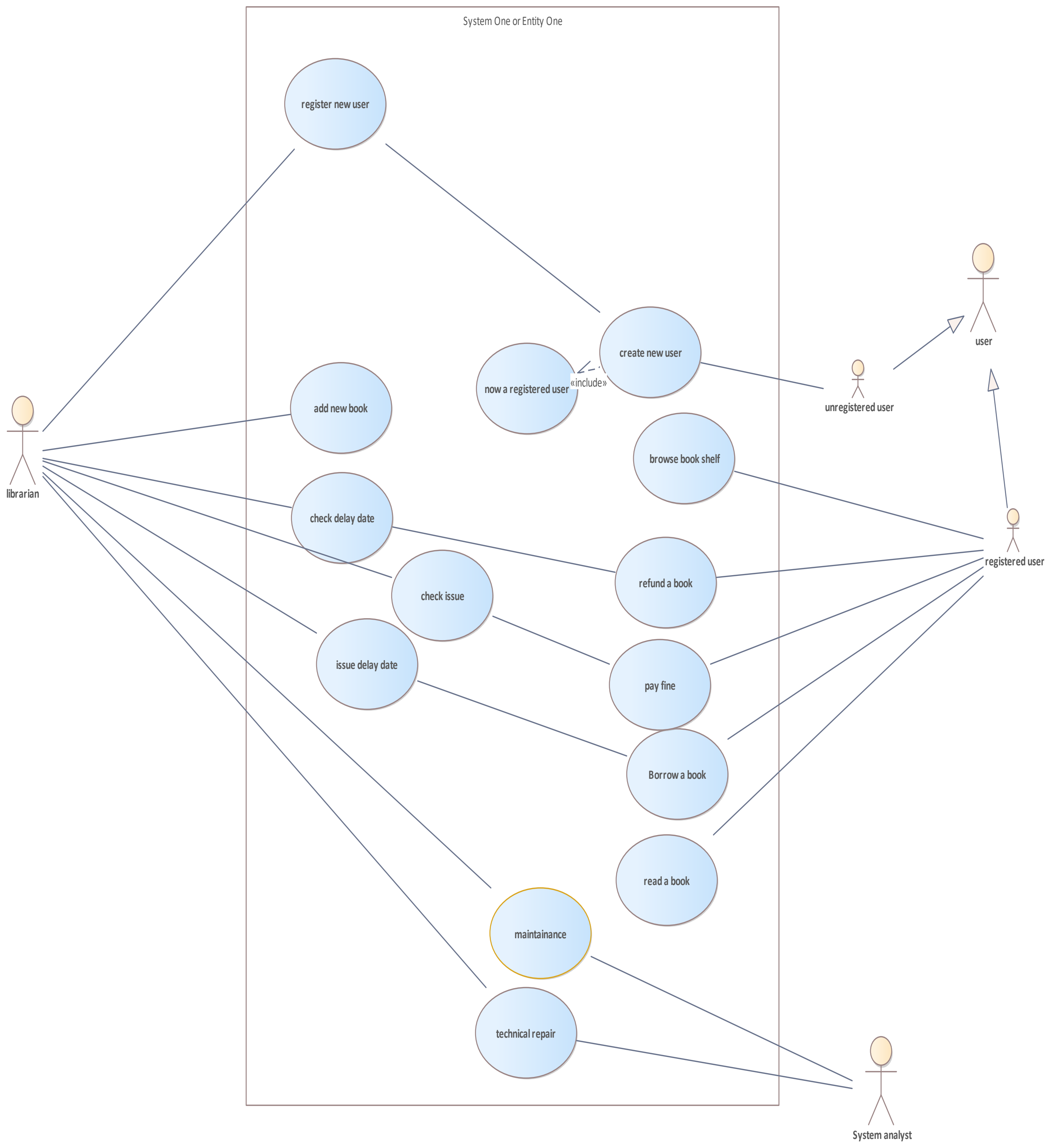
❖ SYSTEM ANALYST REQUIREMENTS

- Constantly makes sure that various parts of the system are are functioning properly.
- Repair any malfunctioning part of the system

- **Use Case Diagram For The LA(Library Automation) SOFTWARE :**

  - The use case diagram is the diagram that summarizes the relationship between actors (, librarian, and the various users generalized as) and the various use cases.

  - The various actors are : **System Analyst**, **Librarian** and the **Users**(Registered and unregistered users).



- **Class Diagram For The LA(Library Automation) SOFTWARE :**

  - Describes the different types of objects and the static relationship that exists between them.

- User Class: It manages all the operations that a user can perform with respect to our Library automation system.
- Book Class: The book class manages all the operations a user as well as a Liberian can perform on a book in the library
- Liberian Class: The Liberian class manages all the operations performed by a Liberian in the system.
- Transaction Class: This class enables a Liberian to initiate a payment in the system
- Bill Class: This class enables a Liberian to create a document containing the amount of a fine paid by a specific user

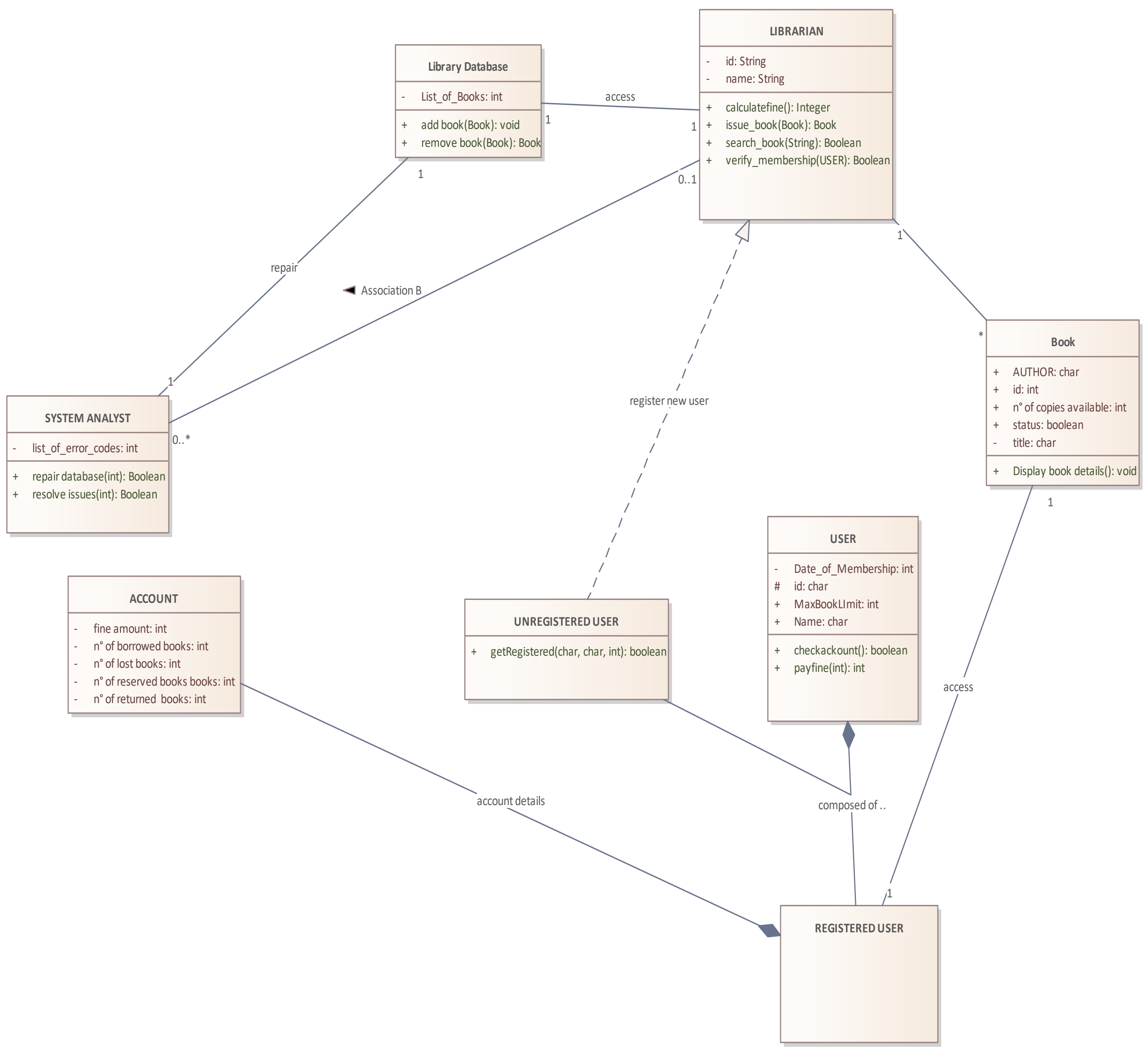
Attributes of the Library automation

- User Class: Name, telnumber, password, Address, DateofMembership, MaxBookLimit, UserId.
- Book Class: Title, Author, Bookid, Status, publication_date.
- Liberian Class: Name, password
- Transaction Class: Transid, Name, Userid, date_of_trans, Book_id, Book_title
- Bill Class: BillNumber, Userid, amount.

Methods of the Library Automation

- User Class: Verify(), search_Book(), payBill().
- Book Class: DisplayBookDetails(), UpdateStatus()

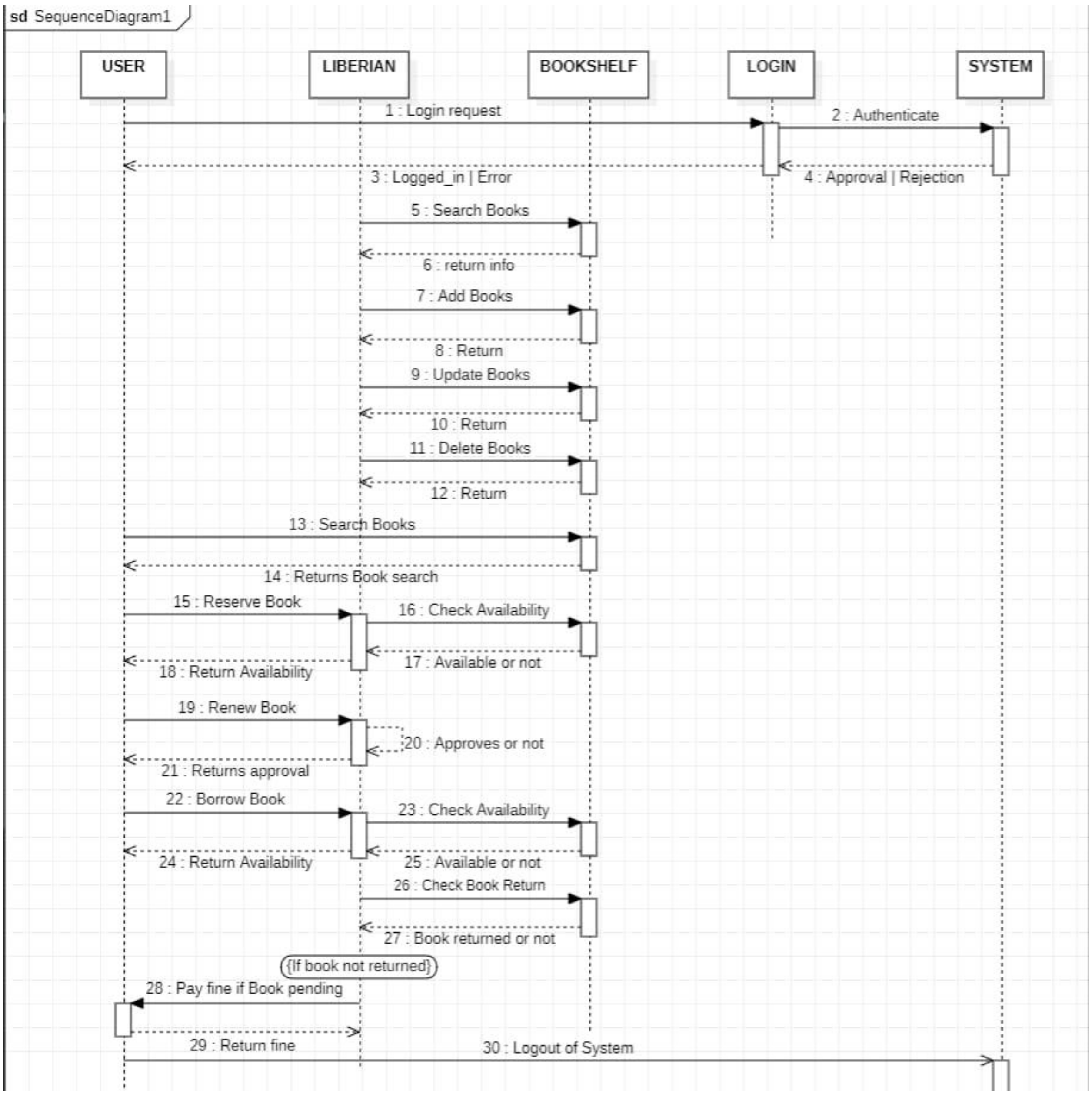
- Liberian Class: Search_Book(), VerifyMembership(), issueBook(), calculateFine(), createBill().
- Transaction Class: createtransaction(), DeleteTransaction()
- Bill Class: BillCreate(), BillUpdate()



LIBRARIAN
- id: String
- name: String
+ calculatefine(): Integer
+ issue_book(Book): Book
+ search_book(String): Boolean
+ verify_membership(USER): Boolean
Library Database
- List_of_Books: int
+ add book(Book): void
+ remove book(Book): Book
access
1
1
1
0..1
repair
Association B
1
0..*
SYSTEM ANALYST
- list_of_error_codes: int
+ repair database(int): Boolean
+ resolve issues(int): Boolean
register new user
1
*
Book
+ AUTHOR: char
+ id: int
+ n° of copies available: int
+ status: boolean
- title: char
+ Display book details(): void
1
USER
- Date_of_Membership: int
# id: char
+ MaxBookLImit: int
+ Name: char
+ checkackount(): boolean
+ payfine(int): int
ACCOUNT
- fine amount: int
- n° of borrowed books: int
- n° of lost books: int
- n° of reserved books books: int
- n° of returned books: int
UNREGISTERED USER
+ getRegistered(char, char, int): boolean
access
account details
composed of ..
1
REGISTERED USER

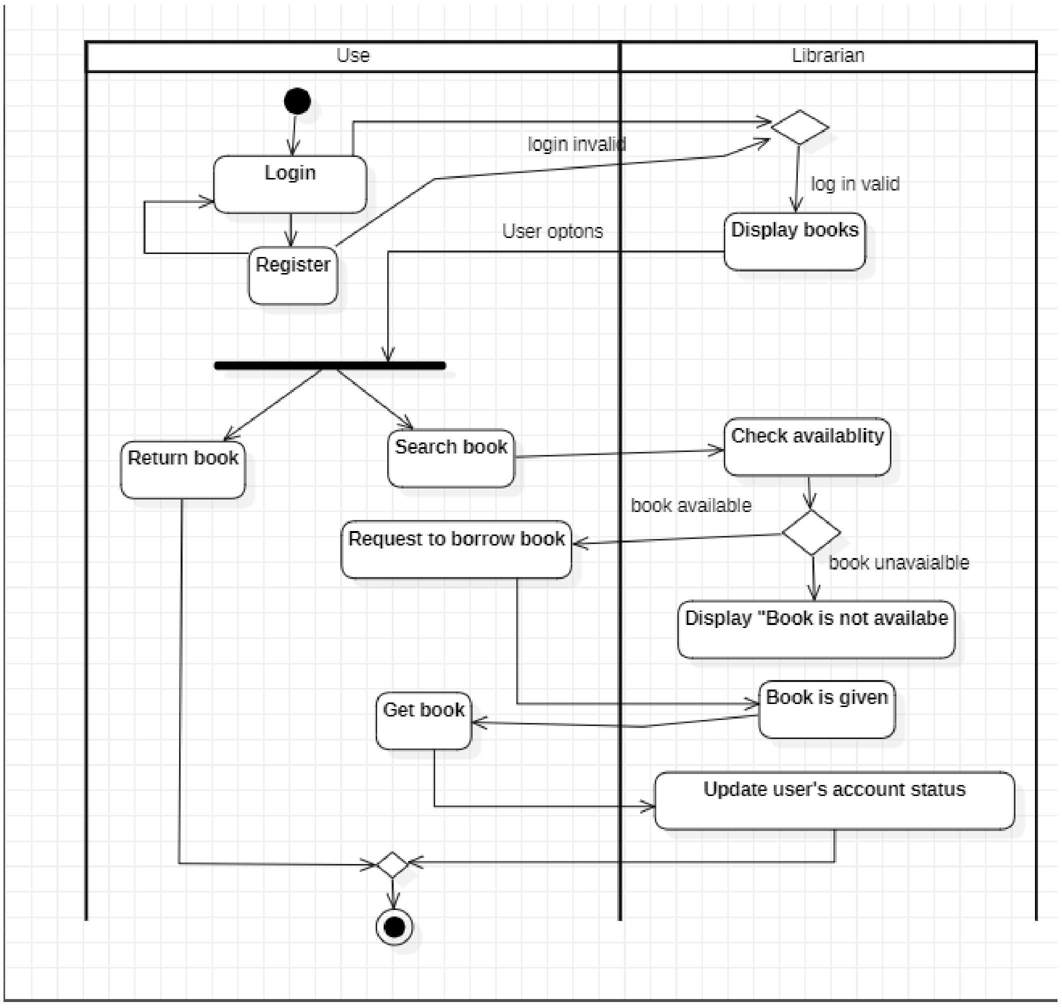
Sequence Diagram For The LA(Library Automation) SOFTWARE :

- Sequence diagrams detail how operations are perfomed.
- Capture interaction between objects in the context of collaboration.



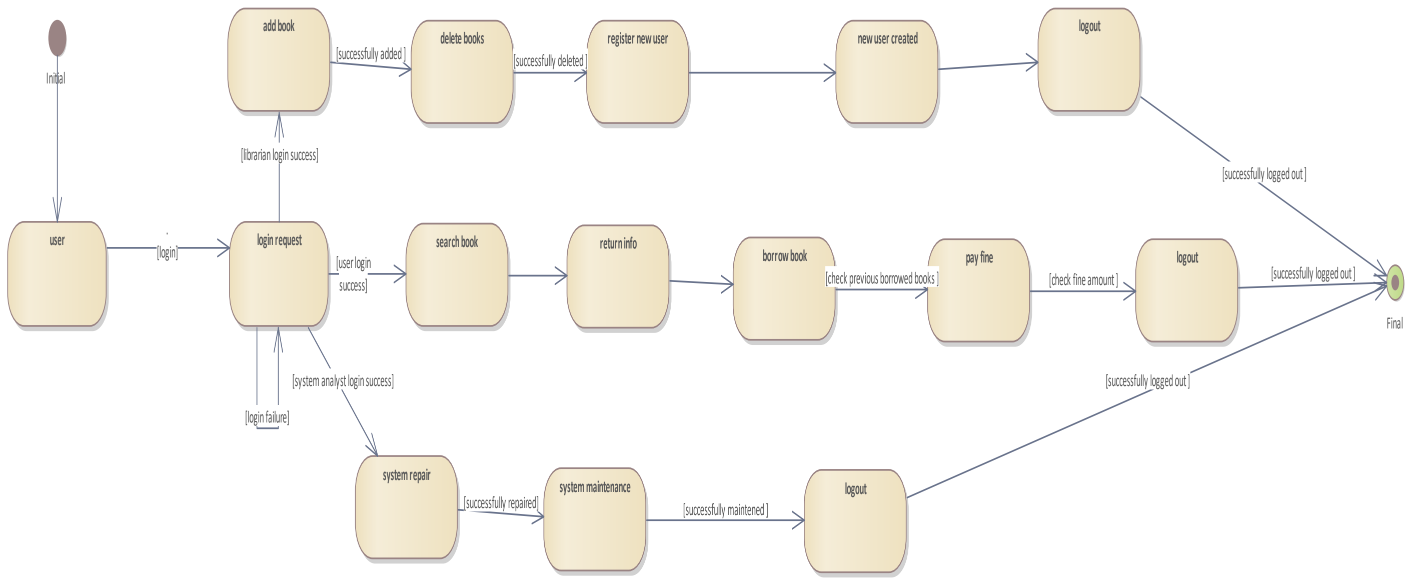
Activity Diagram For The LA(Library Automation) SOFTWARE :

- Actitivity Diagram models the transition from one actitiy to another.

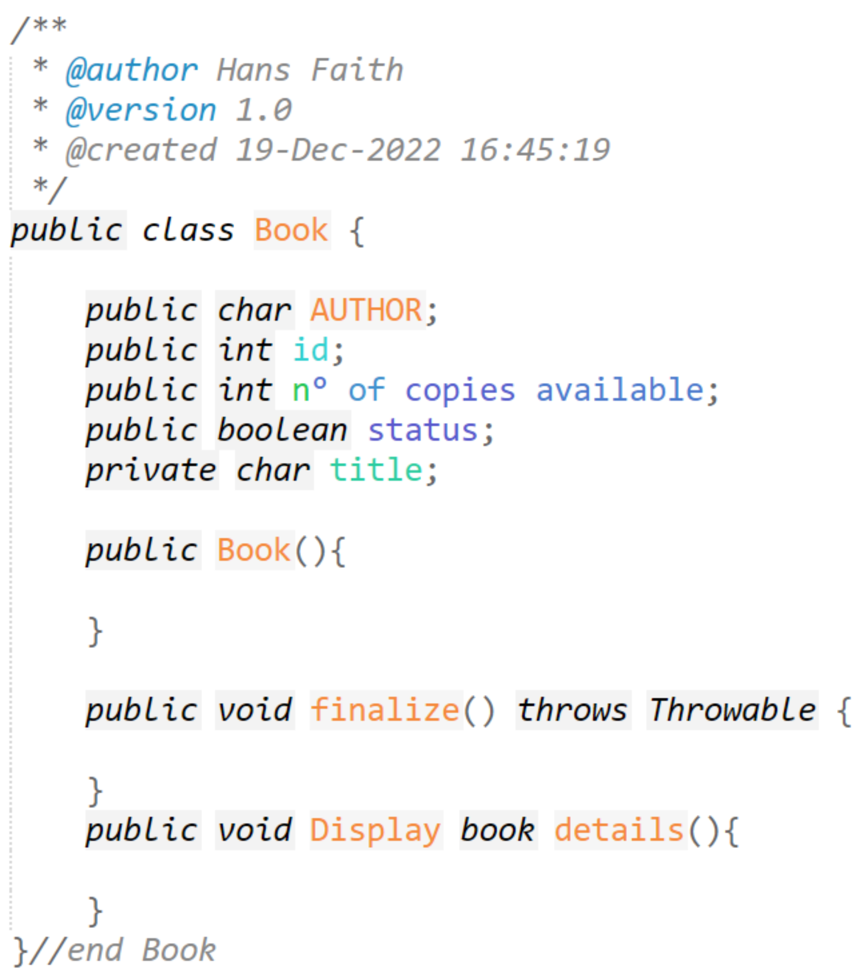


State Diagram For The LA(Library Automation) SOFTWARE :

- State machine diagram shows the various states of a system and how the states react to various events.
- State machine describes all events in relation with a single object.

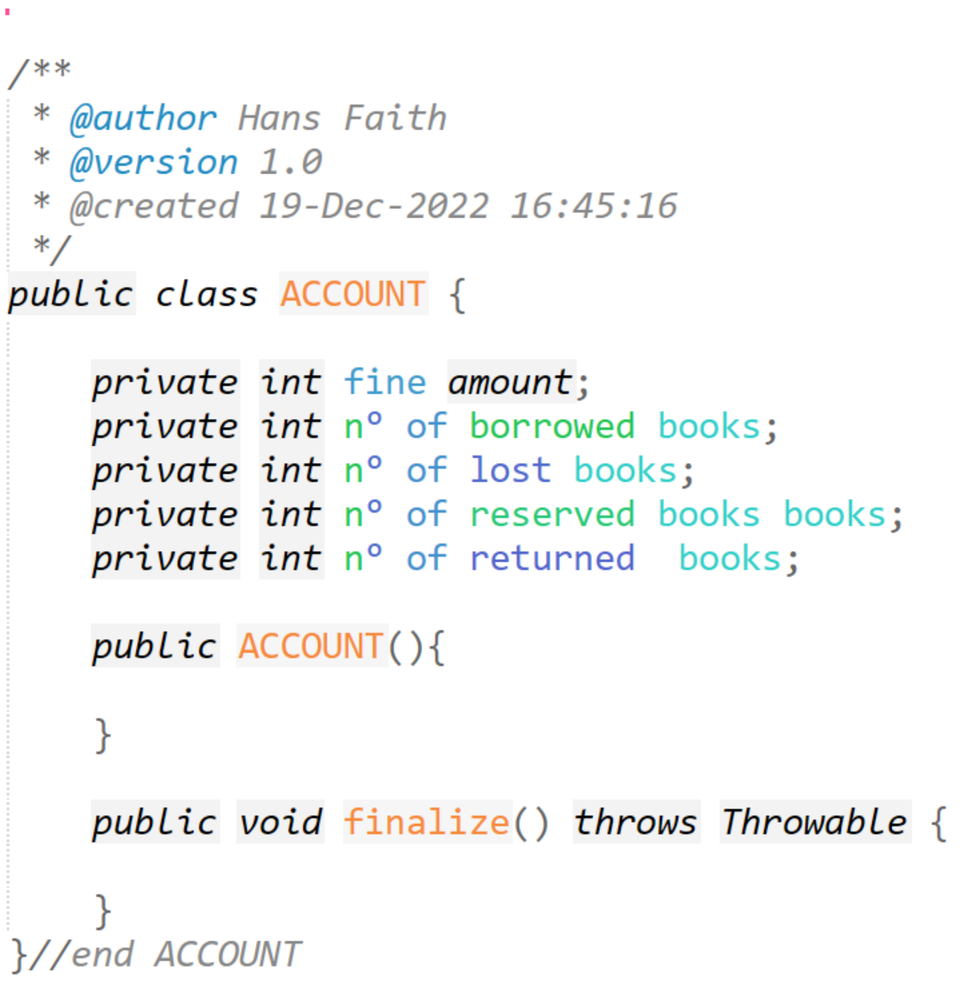


Java Code for Book Class



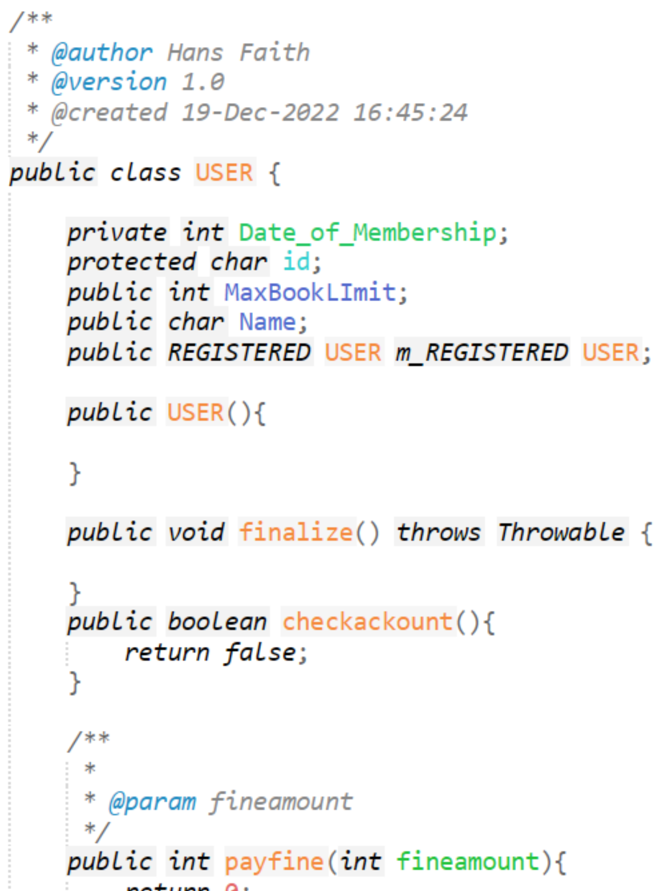
```
/**
 * @author Hans Faith
 * @version 1.0
 * @created 19-Dec-2022 16:45:19
 */
public class Book {

    public char AUTHOR;
    public int id;
    public int n° of copies available;
    public boolean status;
    private char title;

    public Book(){

    }

    public void finalize() throws Throwable {

    }
    public void Display book details(){

    }
}//end Book
```

Java Code for Account Class



```
/**
 * @author Hans Faith
 * @version 1.0
 * @created 19-Dec-2022 16:45:16
 */
public class ACCOUNT {

    private int fine amount;
    private int n° of borrowed books;
    private int n° of lost books;
    private int n° of reserved books books;
    private int n° of returned  books;

    public ACCOUNT(){

    }

    public void finalize() throws Throwable {

    }
}//end ACCOUNT
```

Java Code for User Class



```
/**
 * @author Hans Faith
 * @version 1.0
 * @created 19-Dec-2022 16:45:24
 */
public class USER {

    private int Date_of_Membership;
    protected char id;
    public int MaxBookLImit;
    public char Name;
    public REGISTERED USER m_REGISTERED USER;

    public USER(){

    }

    public void finalize() throws Throwable {

    }
    public boolean checkackount(){
        return false;
    }

    /**
     *
     * @param fineamount
     */
    public int payfine(int fineamount){
        return 0;
```