[Book.java]

```
package LMS;
import java.io.*;
import java.time.temporal.ChronoUnit;
import java.util.*;
public class Book {
    private int bookID;                 // ID given by a library to a book to make it distinguishable from other books
    private String title;           // Title of a book
    private String subject;         // Subject to which a book is related!
    private String author;          // Author of book!
    private boolean isIssued;           // this will be true if the book is currently issued to some borrower.
    private HoldRequestOperations holdRequestsOperations =new HoldRequestOperations();
    static int currentIdNumber =0;      //This will be unique for every book, since it will be incremented when everytime
                                        //when a book is created


    public Book(int id,String t, String s, String a, boolean issued)      // Parameterise cons.
    {
        currentIdNumber++;
        if(id==-1)
        {
            bookID = currentIdNumber;
        }
        else
            bookID=id;

        title = t;
        subject = s;
        author = a;
        isIssued = issued;
    }
    // printing all hold req on a book.
    public void printHoldRequests()
    {
        if (!holdRequestsOperations.holdRequests.isEmpty())
        {
            System.out.println("₩nHold Requests are: ");


System.out.println("-----------------------------------------------------
-------------------------------------------------------------------
--------------");
            System.out.println("No.₩t₩tBook's                    Title₩t₩t₩tBorrower's Name₩t₩t₩tRequest Date");

System.out.println("-----------------------------------------------------
-------------------------------------------------------------------
--------------");
```

```
            for (int i =0; i < holdRequestsOperations.holdRequests.size(); i++)
            {
                System.out.print(i +"-"+"₩t₩t");
                holdRequestsOperations.holdRequests.get(i).print();
            }
        }
        else
            System.out.println("₩nNo Hold Requests.");
    }

    // printing book's Info
    public void printInfo()
    {
        System.out.println(title +"₩t₩t₩t"+ author +"₩t₩t₩t"+ subject);
    }

    // changign Info of a Book
    public void changeBookInfo() throws IOException
    {
        Scanner scanner =new Scanner(System.in);
        String input;

        BufferedReader reader =new BufferedReader(new
InputStreamReader(System.in));

        System.out.println("₩nUpdate Author? (y/n)");
        input = scanner.next();

        if(input.equals("y"))
        {
            System.out.println("₩nEnter new Author: ");
            author = reader.readLine();
        }
        System.out.println("₩nUpdate Subject? (y/n)");
        input = scanner.next();

        if(input.equals("y"))
        {
            System.out.println("₩nEnter new Subject: ");
            subject = reader.readLine();
        }
        System.out.println("₩nUpdate Title? (y/n)");
        input = scanner.next();

        if(input.equals("y"))
        {
            System.out.println("₩nEnter new Title: ");
            title = reader.readLine();
        }

        System.out.println("₩nBook is successfully updated.");

    }
```

```
    /*------------Getter FUNCs.---------*/

    public String getTitle()
    {
        return title;
    }
    public String getSubject()
    {
        return subject;
    }
    public String getAuthor()
    {
        return author;
    }

    public boolean getIssuedStatus()
    {
        return isIssued;
    }

    public void setIssuedStatus(boolean s)
    {
        isIssued = s;
    }

     public int getID()
    {
        return bookID;
    }

     public ArrayList<HoldRequest> getHoldRequests()
    {
        return holdRequestsOperations.holdRequests;
    }
    /*-----------------------------------*/

    // Setter Static Func.
    public static void setIDCount(int n)
    {
        currentIdNumber = n;
    }



//-----------------------------------------------------------------------
-//

    // Placing book on Hold
    public void placeBookOnHold(Borrower bor)
    {
        HoldRequest hr =new HoldRequest(bor,this, new Date());
        holdRequestsOperations.addHoldRequest(hr);          //Add this hold request to
```

```
holdRequests queue of this book
        bor.addHoldRequest(hr);             //Add this hold request to that particular
borrower's class as well

        System.out.println("₩nThe book "+ title +" has been successfully placed on
hold by borrower "+ bor.getName() +".₩n");
    }


   // Request for Holding a Book
    public void makeHoldRequest(Borrower borrower)
    {
        boolean makeRequest =true;
        //If that borrower has already borrowed that particular book. Then he isn't
allowed to make request for that book. He will have to renew the issued book in
order to extend the return deadline.
        for(int i=0;i<borrower.getBorrowedBooks().size();i++)
        {
            if(borrower.getBorrowedBooks().get(i).getBook()==this)
            {
                System.out.println("₩n"+"You have already borrowed "+ title);
                return;
            }
        }


        //If that borrower has already requested for that particular book. Then he isn't
allowed to make the same request again.
        for (int i =0; i < holdRequestsOperations.holdRequests.size(); i++)
        {
            if        ((holdRequestsOperations.holdRequests.get(i).getBorrower()        ==
borrower))
            {
                makeRequest =false;
                break;
            }
        }
        if (makeRequest)
        {
            placeBookOnHold(borrower);
        }
        else
            System.out.println("₩nYou already have one hold request for this
book.₩n");
    }

    // Getting Info of a Hold Request
    public void serviceHoldRequest(HoldRequest hr)
    {
        holdRequestsOperations.removeHoldRequest();
        hr.getBorrower().removeHoldRequest(hr);
    }
```

```
    // Issuing a Book
    public void issueBook(Borrower borrower, Staff staff)
    {
        //First deleting the expired hold requests
        Date today =new Date();

        ArrayList<HoldRequest> hRequests = holdRequestsOperations.holdRequests;

        for (int i =0; i < hRequests.size(); i++)
        {
            HoldRequest hr = hRequests.get(i);

            //Remove that hold request which has expired
            long days = ChronoUnit.DAYS.between(today.toInstant(),
hr.getRequestDate().toInstant());
            days =0-days;

            if(days>Library.getInstance().getHoldRequestExpiry())
            {
                holdRequestsOperations.removeHoldRequest();
                hr.getBorrower().removeHoldRequest(hr);
            }
        }

        if (isIssued)
        {
            System.out.println("₩nThe book "+ title +" is already issued.");
            System.out.println("Would you like to place the book on hold? (y/n)");

            Scanner sc =new Scanner(System.in);
            String choice = sc.next();

            if (choice.equals("y"))
            {
                makeHoldRequest(borrower);
            }
        }

        else
        {
            if (!holdRequestsOperations.holdRequests.isEmpty())
            {
                boolean hasRequest =false;

                for (int i =0; i < holdRequestsOperations.holdRequests.size()
&&!hasRequest;i++)
                {
                    if (holdRequestsOperations.holdRequests.get(i).getBorrower() ==
borrower)
                        hasRequest =true;

                }

                if (hasRequest)
```

```
                {
                    //If this particular borrower has the earliest request for this book
                    if  (holdRequestsOperations.holdRequests.get(0).getBorrower()  ==
borrower)

serviceHoldRequest(holdRequestsOperations.holdRequests.get(0));
                    else
                    {
                        System.out.println("₩nSorry some other users have requested
for this book earlier than you. So you have to wait until their hold requests are
processed.");
                        return;
                    }
                }
                else
                {
                    System.out.println("₩nSome users have already placed this book
on request and you haven't, so the book can't be issued to you.");

                    System.out.println("Would you like to place the book on hold?
(y/n)");
                    Scanner sc =new Scanner(System.in);
                    String choice = sc.next();

                    if (choice.equals("y"))
                    {
                        makeHoldRequest(borrower);
                    }

                    return;
                }
            }

            //If there are no hold requests for this book, then simply issue the book.

            setIssuedStatus(true);

            Loan iHistory =new Loan(borrower,this,staff,null,new Date(),null,false);

            Library.getInstance().addLoan(iHistory);
            borrower.addBorrowedBook(iHistory);

            System.out.println("₩nThe book "+ title +" is successfully issued to "+
borrower.getName() +".");
            System.out.println("₩nIssued by: "+ staff.getName());
        }
    }


    // Returning a Book
    public void returnBook(Borrower borrower, Loan l, Staff staff)
    {
        l.getBook().setIssuedStatus(false);
        l.setReturnedDate(new Date());
```

```
            l.setReceiver(staff);

            borrower.removeBorrowedBook(l);

            l.payFine();

            System.out.println("₩nThe book "+ l.getBook().getTitle() +" is successfully
returned by "+ borrower.getName() +".");
            System.out.println("₩nReceived by: "+ staff.getName());
        }

}    // Book Class Closed
```

//

[Borrower.java]

```
package LMS;
import java.io.*;
import java.util.*;
public class Borrower extends Person
{
    private ArrayList<Loan> borrowedBooks;                    //Those books which are currently borrowed by this borrower
    private ArrayList<HoldRequest> onHoldBooks;   //Those books which are currently requested by this borrower to be on hold


    public Borrower(int id,String name, String address, int phoneNum) // para. cons
    {
        super(id,name,address,phoneNum);

        borrowedBooks =new ArrayList();
        onHoldBooks =new ArrayList();
    }

    // Printing Borrower's Info
    @Override
    public void printInfo()
    {
        super.printInfo();

        printBorrowedBooks();
        printOnHoldBooks();
    }

    // Printing Book's Info Borrowed by Borrower
    public void printBorrowedBooks()
    {
        if (!borrowedBooks.isEmpty())
        {
            System.out.println("₩nBorrowed Books are: ");


System.out.println("-------------------------------------------------------------------------------");
            System.out.println("No.₩t₩tTitle₩t₩t₩tAuthor₩t₩t₩tSubject");

System.out.println("-------------------------------------------------------------------------------");

            for (int i =0; i < borrowedBooks.size(); i++)
            {
                System.out.print(i +"-"+"₩t₩t");
                borrowedBooks.get(i).getBook().printInfo();
                System.out.print("₩n");
            }
        }
```

```
        else
            System.out.println("₩nNo borrowed books.");
    }

    // Printing Book's Info kept on Hold by Borrower
    public void printOnHoldBooks()
    {
        if (!onHoldBooks.isEmpty())
        {
            System.out.println("₩nOn Hold Books are: ");


System.out.println("-----------------------------------------------------
-------------------------");
            System.out.println("No.₩t₩tTitle₩t₩t₩tAuthor₩t₩t₩tSubject");

System.out.println("-----------------------------------------------------
-------------------------");

            for (int i =0; i < onHoldBooks.size(); i++)
            {
                System.out.print(i +"-"+"₩t₩t");
                onHoldBooks.get(i).getBook().printInfo();
                System.out.print("₩n");
            }
        }
        else
            System.out.println("₩nNo On Hold books.");
    }

    // Updating Borrower's Info
    public void updateBorrowerInfo() throws IOException
    {
        String choice;

        Scanner sc =new Scanner(System.in);
        BufferedReader reader =new BufferedReader(new
InputStreamReader(System.in));


        System.out.println("₩nDo you want to update "+ getName() +"'s Name ?
(y/n)");
        choice = sc.next();
        updateBorrowerName(choice, reader);
        System.out.println("₩nDo you want to update "+ getName() +"'s Address ?
(y/n)");
        choice = sc.next();
        updateBorrowerAddress(choice, reader);
        System.out.println("₩nDo you want to update "+ getName() +"'s Phone
Number ? (y/n)");
        choice = sc.next();
        updateBorrowerPhoneNumber(choice, sc);
        System.out.println("₩nBorrower is successfully updated.");
```

```
    }
    private void updateBorrowerPhoneNumber(String choice, Scanner sc) {
        if(choice.equals("y"))
        {
            System.out.println("₩nType New Phone Number: ");
            setPhone(sc.nextInt());
            System.out.println("₩nThe phone number is successfully updated.");
        }
    }
    private void updateBorrowerAddress(String choice, BufferedReader reader) throws
IOException {
        if(choice.equals("y"))
        {
            System.out.println("₩nType New Address: ");
            setAddress(reader.readLine());
            System.out.println("₩nThe address is successfully updated.");
        }
    }
    private void updateBorrowerName(String choice, BufferedReader reader) throws
IOException {
        if(choice.equals("y"))
        {
            System.out.println("₩nType New Name: ");
            setName(reader.readLine());
            System.out.println("₩nThe name is successfully updated.");
        }
    }
    /*-- Adding and Removing from Borrowed Books---*/
    public void addBorrowedBook(Loan iBook)
    {
        borrowedBooks.add(iBook);
    }

    public void removeBorrowedBook(Loan iBook)
    {
        borrowedBooks.remove(iBook);
    }

    /*-----------------------------------------*/

    /*-- Adding and Removing from On Hold Books---*/
    public void addHoldRequest(HoldRequest hr)
    {
        onHoldBooks.add(hr);
    }

    public void removeHoldRequest(HoldRequest hr)
    {
        onHoldBooks.remove(hr);
    }

    /*-----------------------------------------*/

    /*-----------Getter FUNCs. ------------------*/
```

```
    public ArrayList<Loan> getBorrowedBooks()
    {
        return borrowedBooks;
    }

    public ArrayList<HoldRequest> getOnHoldBooks()
    {
        return onHoldBooks;
    }
    /*-----------------------------------------*/
}
```

//

[Clerk.java]

```
package LMS;
public class Clerk extends Staff {

    int deskNo;       //Desk Number of the Clerk
    public static int currentdeskNumber =0;

    public Clerk(int id, String n, String a,int ph, double s,int dk) // para cons.
    {
        super(id,n,a,ph,s);

        if(dk ==-1)
        {
            deskNo = currentdeskNumber;
        }
        else
        {
            deskNo=dk;
        }

        currentdeskNumber++;
    }

    // Printing Clerk's Info
    @Override
    public void printInfo()
    {
        super.printInfo();
        System.out.println("Desk Number: "+ deskNo);
    }

}    // Clerk's Class Closed
```

//

[HoldRequest.java]

```
package LMS;
import java.util.Date;
public class HoldRequest {

    Borrower borrower;
    Book book;
    Date requestDate;

    public HoldRequest(Borrower bor, Book b, Date reqDate)  // para cons.
    {
        borrower = bor;
        book = b;
        requestDate = reqDate;
    }

    /*----- Getter FUNCs.--------*/
    public Borrower getBorrower()
    {
        return borrower;
    }

    public Book getBook()
    {
        return book;
    }

    public Date getRequestDate()
    {
        return requestDate;
    }
    /*-------------------------*/

    // Print Hold Request Info
    public void print()
    {
        System.out.print(book.getTitle()        +"₩t₩t₩t₩t"+        borrower.getName()
+"₩t₩t₩t₩t"   + requestDate +"₩n");
    }
}// HoldRequest Class Closed
```

//

[HoldRequestOperations.java]

```
package LMS;
import java.util.ArrayList;
public class HoldRequestOperations {
    static ArrayList <HoldRequest> holdRequests;
     public HoldRequestOperations()
     {
          holdRequests=new ArrayList<>();
     }
     // adding a hold req.
     public void addHoldRequest(HoldRequest hr)
     {
          holdRequests.add(hr);
     }
     // removing a hold req.
     public void removeHoldRequest()
     {
          if(!holdRequests.isEmpty())
          {
               holdRequests.remove(0);
          }
     }
}
```

//

[Librarian.java]

```
package LMS;
import static LMS.Library.librarian;
import static LMS.Library.persons;
public class Librarian extends Staff {
    int officeNo;       //Office Number of the Librarian
    public static int currentOfficeNumber =0;
    public Librarian(int id, String n, String a, int p, double s, int of) // para cons.
    {
        super(id, n, a, p, s);
        if (of ==-1)
            officeNo = currentOfficeNumber;
        else
            officeNo = of;
        currentOfficeNumber++;
    }
    // Printing Librarian's Info
    @Override
    public void printInfo() {
        super.printInfo();
        System.out.println("Office Number: "+ officeNo);
    }
    public static boolean addLibrarian(Librarian lib) {
        //One Library can have only one Librarian
        if (librarian ==null) {
            librarian = lib;
            persons.add(librarian);
            return true;
        } else
            System.out.println("₩nSorry, the library already has one librarian. New
Librarian can't be created.");
        return false;
    }
}
```

//

[Library.java]

```
package LMS;
// Including Header Files.
import java.io.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.SQLIntegrityConstraintViolationException;
import java.sql.Statement;
import java.sql.Types;
import java.util.*;
import java.util.logging.Level;
import java.util.logging.Logger;
public class Library {

    private String name;                                    // name of library
    public static Librarian librarian;                      // object of Librarian (only
one)
    public static ArrayList <Person> persons;               // all clerks and
borrowers
    private ArrayList <Book> booksInLibrary;                // all books in library are
here!

    private ArrayList <Loan> loans;                         // history of all books which
have been issued

    public int book_return_deadline;                        //return deadline after which
fine will be generated each day
    public double per_day_fine;

    public int hold_request_expiry;                         //number of days after which a
hold request will expire
    //Created object of the hold request operations
    private HoldRequestOperations holdRequestsOperations =new
HoldRequestOperations();

    /*----Following Singleton Design Pattern (Lazy Instantiation)-----------*/
    private static Library obj;
    public static Library getInstance()
    {
        if(obj==null)
        {
            obj =new Library();
        }

        return obj;
    }

/*-----------------------------------------------------------------------
---*/
```

```
private Library()    // default cons.
{
    name =null;
    librarian =null;
    persons =new ArrayList();

    booksInLibrary =new ArrayList();
    loans =new ArrayList();
}

/*------------Setter FUNCs.------------*/

public void setReturnDeadline(int deadline)
{
    book_return_deadline = deadline;
}
public void setFine(double perDayFine)
{
    per_day_fine = perDayFine;
}
public void setRequestExpiry(int hrExpiry)
{
    hold_request_expiry = hrExpiry;
}
/*-------------------------------------*/


// Setter Func.
public void setName(String n)
{
    name = n;
}

/*-----------Getter FUNCs.------------*/

public int getHoldRequestExpiry()
{
    return hold_request_expiry;
}

public ArrayList<Person> getPersons()
{
    return persons;
}

public Librarian getLibrarian()
{
    return librarian;
}

public String getLibraryName()
{
```

```
        return name;
    }
    public ArrayList<Book> getBooks()
    {
        return booksInLibrary;
    }

    /*----------------------------------------*/
    /*-----Adding other People in Library----*/
    public void addClerk(Clerk c)
    {
        persons.add(c);
    }
    public void addBorrower(Borrower b)
    {
        persons.add(b);
    }

    public void addLoan(Loan l)
    {
        loans.add(l);
    }

    /*----------------------------------------------*/

    /*-----------Finding People in Library-------------*/
    public Borrower findBorrower()
    {
        System.out.println("₩nEnter Borrower's ID: ");

        int id =0;

        Scanner scanner =new Scanner(System.in);

        try{
            id = scanner.nextInt();
        }
        catch (java.util.InputMismatchException e)
        {
            System.out.println("₩nInvalid Input");
        }
        for (int i =0; i < persons.size(); i++)
        {
            if (persons.get(i).getID() == id &&
persons.get(i).getClass().getSimpleName().equals("Borrower"))
                return (Borrower)(persons.get(i));
        }

        System.out.println("₩nSorry this ID didn't match any Borrower's ID.");
        return null;
    }

    public Clerk findClerk()
    {
```

```
        System.out.println("₩nEnter Clerk's ID: ");

        int id =0;

        Scanner scanner =new Scanner(System.in);

        try{
            id = scanner.nextInt();
        }
        catch (java.util.InputMismatchException e)
        {
            System.out.println("₩nInvalid Input");
        }
        for (int i =0; i < persons.size(); i++)
        {
            if (persons.get(i).getID() == id &&
persons.get(i).getClass().getSimpleName().equals("Clerk"))
                return (Clerk)(persons.get(i));
        }

        System.out.println("₩nSorry this ID didn't match any Clerk's ID.");
        return null;
    }

    /*-------- FUNCS. on Books In Library---------------*/
    public void addBookinLibrary(Book b)
    {
        booksInLibrary.add(b);
    }

    //When this function is called, only the pointer of the book placed in
booksInLibrary is removed. But the real object of book
    //is still there in memory because pointers of that book placed in IssuedBooks
and ReturnedBooks are still pointing to that book. And we
    //are maintaining those pointers so that we can maintain history.
    //But if we donot want to maintain history then we can delete those pointers
placed in IssuedBooks and ReturnedBooks as well which are
    //pointing to that book. In this way the book will be really removed from memory.
    public void removeBookfromLibrary(Book b)
    {
        boolean delete =true;

        //Checking if this book is currently borrowed by some borrower
        for (int i =0; i < persons.size() && delete; i++)
        {
            if (persons.get(i).getClass().getSimpleName().equals("Borrower"))
            {
                ArrayList<Loan> borBooks =
((Borrower)(persons.get(i))).getBorrowedBooks();

                for (int j =0; j < borBooks.size() && delete; j++)
                {
                    if (borBooks.get(j).getBook() == b)
                    {
```

```
                        delete =false;
                        System.out.println("This particular book is currently borrowed by some borrower.");
                    }
                }
            }
        }

        if (delete)
        {
            System.out.println("₩nCurrently this book is not borrowed by anyone.");
            ArrayList<HoldRequest> hRequests = b.getHoldRequests();

            if(!hRequests.isEmpty())
            {
                System.out.println("₩nThis book might be on hold requests by some borrowers. Deleting this book will delete the relevant hold requests too.");
                System.out.println("Do you still want to delete the book? (y/n)");

                Scanner sc =new Scanner(System.in);

                while (true)
                {
                    String choice = sc.next();

                    if(choice.equals("y") || choice.equals("n"))
                    {
                        if(choice.equals("n"))
                        {
                            System.out.println("₩nDelete Unsuccessful.");
                            return;
                        }
                        else
                        {
                            //Empty the books hold request array
                            //Delete the hold request from the borrowers too
                            for (int i =0; i < hRequests.size() && delete; i++)
                            {
                                HoldRequest hr = hRequests.get(i);
                                hr.getBorrower().removeHoldRequest(hr);
                                holdRequestsOperations.removeHoldRequest();
                            }
                        }
                    }
                    else
                        System.out.println("Invalid Input. Enter (y/n): ");
                }

            }
            else
                System.out.println("This book has no hold requests.");

            booksInLibrary.remove(b);
            System.out.println("The book is successfully removed.");
```

```
        }
        else
            System.out.println("₩nDelete Unsuccessful.");
    }


    // Searching Books on basis of title, Subject or Author
    public ArrayList<Book> searchForBooks() throws IOException
    {
        String choice;
        String title ="", subject ="", author ="";

        Scanner sc =new Scanner(System.in);
        BufferedReader reader =new BufferedReader(new
InputStreamReader(System.in));

        while (true)
        {
            System.out.println("₩nEnter either '1' or '2' or '3' for search by Title,
Subject or Author of Book respectively: ");
            choice = sc.next();

            if (choice.equals("1") || choice.equals("2") || choice.equals("3"))
                break;
            else
                System.out.println("₩nWrong Input!");
        }
        if (choice.equals("1"))
        {
            System.out.println("₩nEnter the Title of the Book: ");
            title = reader.readLine();
        }
        else if (choice.equals("2"))
        {
            System.out.println("₩nEnter the Subject of the Book: ");
            subject = reader.readLine();
        }

        else
        {
            System.out.println("₩nEnter the Author of the Book: ");
            author = reader.readLine();
        }

        ArrayList<Book> matchedBooks =new ArrayList();

        //Retrieving all the books which matched the user's search query
        for(int i =0; i < booksInLibrary.size(); i++)
        {
            Book b = booksInLibrary.get(i);

            if (choice.equals("1"))
            {
```

```
                if (b.getTitle().equals(title))
                    matchedBooks.add(b);
            }
            else if (choice.equals("2"))
            {
                if (b.getSubject().equals(subject))
                    matchedBooks.add(b);
            }
            else
            {
                if (b.getAuthor().equals(author))
                    matchedBooks.add(b);
            }
        }

        //Printing all the matched Books
        if (!matchedBooks.isEmpty())
        {
            System.out.println("₩nThese books are found: ₩n");


System.out.println("-----------------------------------------------------
-------------------------");
            System.out.println("No.₩t₩tTitle₩t₩t₩tAuthor₩t₩t₩tSubject");

System.out.println("-----------------------------------------------------
-------------------------");

            for (int i =0; i < matchedBooks.size(); i++)
            {
                System.out.print(i +"-"+"₩t₩t");
                matchedBooks.get(i).printInfo();
                System.out.print("₩n");
            }

            return matchedBooks;
        }
        else
        {
            System.out.println("₩nSorry. No Books were found related to your
query.");
            return null;
        }
    }



    // View Info of all Books in Library
     public void viewAllBooks()
    {
        if (!booksInLibrary.isEmpty())
        {
            System.out.println("₩nBooks are: ");
```

```
System.out.println("-----------------------------------------------------
-------------------------");
            System.out.println("No.₩t₩tTitle₩t₩t₩tAuthor₩t₩t₩tSubject");

System.out.println("-----------------------------------------------------
-------------------------");

            for (int i =0; i < booksInLibrary.size(); i++)
            {
                System.out.print(i +"-"+"₩t₩t");
                booksInLibrary.get(i).printInfo();
                System.out.print("₩n");
            }
        }
        else
            System.out.println("₩nCurrently, Library has no books.");
    }

    //Computes total fine for all loans of a borrower
    public double computeFine2(Borrower borrower)
    {

System.out.println("-----------------------------------------------------
-------------------------------------------------------------------------
-------------------------------------------");
        System.out.println("No.₩t₩tBook's    Title₩t₩tBorrower's    Name₩t₩t₩tIssued
Date₩t₩t₩tReturned Date₩t₩t₩t₩tFine(Rs)");

System.out.println("-----------------------------------------------------
-------------------------------------------------------------------------
-----------------------------------------");

        double totalFine =0;
        double per_loan_fine =0;

        for (int i =0; i < loans.size(); i++)
        {
            Loan l = loans.get(i);

            if ((l.getBorrower() == borrower))
            {
                per_loan_fine = l.computeFine1();
                System.out.print(i      +"-"+"₩t₩t"+      loans.get(i).getBook().getTitle()
+"₩t₩t₩t"+ loans.get(i).getBorrower().getName() +"₩t₩t"+ loans.get(i).getIssuedDate()
+     "₩t₩t₩t"+   loans.get(i).getReturnDate()  +"₩t₩t₩t₩t"+   per_loan_fine     +"₩n");


                totalFine += per_loan_fine;
            }
        }

        return totalFine;
    }
```

```
    public void createPerson(char x)
    {
        Scanner sc =new Scanner(System.in);
        BufferedReader reader =new BufferedReader(new
InputStreamReader(System.in));

        System.out.println("₩nEnter Name: ");
        String n ="";
        try {
            n = reader.readLine();
        } catch (IOException ex) {
            Logger.getLogger(Library.class.getName()).log(Level.SEVERE, null, ex);
        }
        System.out.println("Enter Address: ");
        String address ="";
        try {
            address = reader.readLine();
        } catch (IOException ex) {
            Logger.getLogger(Library.class.getName()).log(Level.SEVERE, null, ex);
        }

        int phone =0;

        try{
            System.out.println("Enter Phone Number: ");
            phone = sc.nextInt();
        }
        catch (java.util.InputMismatchException e)
        {
            System.out.println("₩nInvalid Input.");
        }

        //If clerk is to be created
        if (x =='c')
        {
            double salary =0;

            try{
                System.out.println("Enter Salary: ");
                salary = sc.nextDouble();
            }
            catch (java.util.InputMismatchException e)
            {
                System.out.println("₩nInvalid Input.");
            }

            Clerk c =new Clerk(-1,n,address,phone,salary,-1);
            addClerk(c);

            System.out.println("₩nClerk with name "+ n +" created successfully.");
            System.out.println("₩nYour ID is : "+ c.getID());
            System.out.println("Your Password is : "+ c.getPassword());
```

```
        }

        //If librarian is to be created
        else if (x =='l')
        {
            double salary =0;
            try{
                System.out.println("Enter Salary: ");
                salary = sc.nextDouble();
            }
            catch (java.util.InputMismatchException e)
            {
                System.out.println("\nInvalid Input.");
            }

            Librarian l =new Librarian(-1,n,address,phone,salary,-1);
            if(Librarian.addLibrarian(l))
            {
                System.out.println("\nLibrarian with name "+ n +" created successfully.");
                System.out.println("\nYour ID is : "+ l.getID());
                System.out.println("Your Password is : "+ l.getPassword());
            }
        }
        //If borrower is to be created
        else
        {
            Borrower b =new Borrower(-1,n,address,phone);
            addBorrower(b);
            System.out.println("\nBorrower with name "+ n +" created successfully.");
            System.out.println("\nYour ID is : "+ b.getID());
            System.out.println("Your Password is : "+ b.getPassword());
        }
    }


    public void createBook(String title, String subject, String author)
    {
        Book b =new Book(-1,title,subject,author,false);

        addBookinLibrary(b);

        System.out.println("\nBook with Title "+ b.getTitle() +" is successfully created.");
    }


    // Called when want an access to Portal
    public Person login()
    {
        Scanner input =new Scanner(System.in);

        int id =0;
        String password ="";
```

```
        System.out.println("₩nEnter ID: ");

        try{
            id = input.nextInt();
        }
        catch (java.util.InputMismatchException e)
        {
            System.out.println("₩nInvalid Input");
        }

        System.out.println("Enter Password: ");
        password = input.next();

        for (int i =0; i < persons.size(); i++)
        {
            if (persons.get(i).getID() == id &&
persons.get(i).getPassword().equals(password))
            {
                System.out.println("₩nLogin Successful");
                return persons.get(i);
            }
        }

        if(librarian!=null)
        {
            if (librarian.getID() == id && librarian.getPassword().equals(password))
            {
                System.out.println("₩nLogin Successful");
                return librarian;
            }
        }

        System.out.println("₩nSorry! Wrong ID or Password");
        return null;
    }


    // History when a Book was Issued and was Returned!
    public void viewHistory()
    {
        if (!loans.isEmpty())
        {
            System.out.println("₩nIssued Books are: ");


System.out.println("-----------------------------------------------------
-----------------------------------------------------------------------
-----------------------------");
            System.out.println("No.₩tBook's Title₩tBorrower's Name₩t Issuer's
Name₩t₩tIssued Date₩t₩t₩tReceiver's Name₩t₩tReturned Date₩t₩tFine Paid");

System.out.println("-----------------------------------------------------
-----------------------------------------------------------------------
```

```
------------------------------");

            for (int i =0; i < loans.size(); i++)
            {
                if(loans.get(i).getIssuer()!=null)
                    System.out.print(i      +"-"+"₩t"+      loans.get(i).getBook().getTitle()
+"₩t₩t₩t"+                  loans.get(i).getBorrower().getName()                  +"₩t₩t"+
loans.get(i).getIssuer().getName()  +"₩t      "+ loans.get(i).getIssuedDate());

                if (loans.get(i).getReceiver() !=null)
                {
                    System.out.print("₩t"+            loans.get(i).getReceiver().getName()
+"₩t₩t"+ loans.get(i).getReturnDate()  +"₩t     "+ loans.get(i).getFineStatus()  +"₩n");
                }
                else
                    System.out.print("₩t₩t"+"--"+"₩t₩t₩t"+"--"+"₩t₩t"+"--"+"₩n");
            }
        }
        else
            System.out.println("₩nNo issued books.");
    }
```

```
//-------------------------------------------------------------------------
---------------------//
    /*--------------------------------IN-  COLLABORATION  WITH  DATA
BASE-----------------------------------------*/

    // Making Connection With Database
    public Connection makeConnection()
    {
        try
        {
            String host ="jdbc:derby://localhost:1527/LMS";
            String uName ="haris";
            String uPass="123";
            Connection con = DriverManager.getConnection( host, uName, uPass );
            return con;
        }
        catch ( SQLException err )
        {
            System.out.println( err.getMessage( ) );
            return null;
        }
    }
```

```
    // Loading all info in code via Database.
    public void populateLibrary(Connection con) throws SQLException, IOException
    {
            Library lib =this;
            Statement stmt = con.createStatement( );

            /* --- Populating Book ----*/
            String SQL ="SELECT * FROM BOOK";
            ResultSet rs = stmt.executeQuery( SQL );

            if(!rs.next())
            {
               System.out.println("₩nNo Books Found in Library");
            }
            else
            {
                int maxID =0;

                do
                {
                    if(rs.getString("TITLE") !=null && rs.getString("AUTHOR")!=null &&
rs.getString("SUBJECT")!=null && rs.getInt("ID")!=0)
                    {
                        String title=rs.getString("TITLE");
                        String author=rs.getString("AUTHOR");
                        String subject=rs.getString("SUBJECT");
                        int id= rs.getInt("ID");
                        boolean issue=rs.getBoolean("IS_ISSUED");
                        Book b =new Book(id,title,subject,author,issue);
                        addBookinLibrary(b);

                        if (maxID < id)
                            maxID = id;
                    }
                }while(rs.next());

                // setting Book Count
                Book.setIDCount(maxID);
            }

            /* ----Populating Clerks----*/

            SQL = "SELECT ID,PNAME,ADDRESS,PASSWORD,PHONE_NO,SALARY,DESK_NO FROM PERSON INNER
JOIN CLERK ON ID=C_ID INNER JOIN STAFF ON S_ID=C_ID";

            rs=stmt.executeQuery(SQL);

            if(!rs.next())
            {
               System.out.println("No clerks Found in Library");
            }
            else
```

```
{
    do
    {
        int id=rs.getInt("ID");
        String cname=rs.getString("PNAME");
        String adrs=rs.getString("ADDRESS");
        int phn=rs.getInt("PHONE_NO");
        double sal=rs.getDouble("SALARY");
        int desk=rs.getInt("DESK_NO");
        Clerk c =new Clerk(id,cname,adrs,phn,sal,desk);

        addClerk(c);
    }
    while(rs.next());

}

/*-----Populating Librarian---*/
S Q L = " S E L E C T ID,PNAME,ADDRESS,PASSWORD,PHONE_NO,SALARY,OFFICE_NO FROM PERSON INNER JOIN LIBRARIAN ON ID=L_ID INNER JOIN STAFF ON S_ID=L_ID";

rs=stmt.executeQuery(SQL);
if(!rs.next())
{
   System.out.println("No Librarian Found in Library");
}
else
{
    do
    {
        int id=rs.getInt("ID");
        String lname=rs.getString("PNAME");
        String adrs=rs.getString("ADDRESS");
        int phn=rs.getInt("PHONE_NO");
        double sal=rs.getDouble("SALARY");
        int off=rs.getInt("OFFICE_NO");
        Librarian l=new Librarian(id,lname,adrs,phn,sal,off);
        Librarian.addLibrarian(l);

    }while(rs.next());

}

/*---Populating Borrowers (partially)!!!!!!--------*/

SQL="SELECT ID,PNAME,ADDRESS,PASSWORD,PHONE_NO FROM PERSON INNER JOIN BORROWER ON ID=B_ID";

rs=stmt.executeQuery(SQL);

if(!rs.next())
{
   System.out.println("No Borrower Found in Library");
```

```
}
else
{
    do
    {
            int id=rs.getInt("ID");
            String name=rs.getString("PNAME");
            String adrs=rs.getString("ADDRESS");
            int phn=rs.getInt("PHONE_NO");

            Borrower b=new Borrower(id,name,adrs,phn);
            addBorrower(b);

    }while(rs.next());

}

/*----Populating Loan----*/

SQL="SELECT * FROM LOAN";

rs=stmt.executeQuery(SQL);
if(!rs.next())
{
   System.out.println("No Books Issued Yet!");
}
else
{
    do
        {
            int borid=rs.getInt("BORROWER");
            int bokid=rs.getInt("BOOK");
            int iid=rs.getInt("ISSUER");
            Integer rid=(Integer)rs.getObject("RECEIVER");
            int rd=0;
            Date rdate;

            Date idate=new Date (rs.getTimestamp("ISS_DATE").getTime());

            if(rid!=null)      // if there is a receiver
            {
                rdate=new Date (rs.getTimestamp("RET_DATE").getTime());
                rd=(int)rid;
            }
            else
            {
                rdate=null;
            }

            boolean fineStatus = rs.getBoolean("FINE_PAID");

            boolean set=true;

            Borrower bb =null;
```

```
                    for(int i=0;i<getPersons().size() && set;i++)
                    {
                        if(getPersons().get(i).getID()==borid)
                        {
                            set=false;
                            bb=(Borrower)(getPersons().get(i));
                        }
                    }

                    set =true;
                    Staff s[]=new Staff[2];

                    if(iid==getLibrarian().getID())
                    {
                        s[0]=getLibrarian();
                    }

                    else
                    {
                        for(int k=0;k<getPersons().size() && set;k++)
                        {
                            if(getPersons().get(k).getID()==iid                    &&
getPersons().get(k).getClass().getSimpleName().equals("Clerk"))
                            {
                                set=false;
                                s[0]=(Clerk)(getPersons().get(k));
                            }
                        }
                    }

                    set=true;
                    // If not returned yet...
                    if(rid==null)
                    {
                        s[1]=null;   // no reciever
                        rdate=null;
                    }
                    else
                    {
                        if(rd==getLibrarian().getID())
                            s[1]=getLibrarian();
                        else
                        {      //System.out.println("ff");
                            for(int k=0;k<getPersons().size() && set;k++)
                            {
                                if(getPersons().get(k).getID()==rd                    &&
getPersons().get(k).getClass().getSimpleName().equals("Clerk"))
                                {
                                    set=false;
                                    s[1]=(Clerk)(getPersons().get(k));
                                }
                            }
```

```
                        }
                    }

                    set=true;

                    ArrayList<Book> books = getBooks();

                    for(int k=0;k<books.size() && set;k++)
                    {
                        if(books.get(k).getID()==bokid)
                        {
                          set=false;
                          Loan                          l                          =new
Loan(bb,books.get(k),s[0],s[1],idate,rdate,fineStatus);
                          loans.add(l);
                        }
                    }

                }while(rs.next());
            }

            /*----Populationg Hold Books----*/

            SQL="SELECT * FROM ON_HOLD_BOOK";

            rs=stmt.executeQuery(SQL);
            if(!rs.next())
            {
               System.out.println("No Books on Hold Yet!");
            }
            else
            {
                do
                    {
                        int borid=rs.getInt("BORROWER");
                        int bokid=rs.getInt("BOOK");
                        Date off=new Date (rs.getDate("REQ_DATE").getTime());

                        boolean set=true;
                        Borrower bb =null;

                        ArrayList<Person> persons = lib.getPersons();

                        for(int i=0;i<persons.size() && set;i++)
                        {
                            if(persons.get(i).getID()==borid)
                            {
                                set=false;
                                bb=(Borrower)(persons.get(i));
                            }
                        }

                        set=true;
```

```
                        ArrayList<Book> books = lib.getBooks();

                        for(int i=0;i<books.size() && set;i++)
                        {
                            if(books.get(i).getID()==bokid)
                            {
                              set=false;
                               HoldRequest                                    hbook=new
HoldRequest(bb,books.get(i),off);
                               holdRequestsOperations.addHoldRequest(hbook);
                               bb.addHoldRequest(hbook);
                            }
                        }
                        }while(rs.next());
                }

                /* --- Populating Borrower's Remaining Info----*/

                // Borrowed Books
                SQL="SELECT ID,BOOK FROM PERSON INNER JOIN BORROWER ON
ID=B_ID INNER JOIN BORROWED_BOOK ON B_ID=BORROWER ";

                rs=stmt.executeQuery(SQL);

                if(!rs.next())
                {
                   System.out.println("No Borrower has borrowed yet from Library");
                }
                else
                {

                    do
                        {
                            int id=rs.getInt("ID");        // borrower
                            int bid=rs.getInt("BOOK");    // book

                            Borrower bb=null;
                            boolean set=true;
                            boolean okay=true;

                            for(int i=0;i<lib.getPersons().size() && set;i++)
                            {

if(lib.getPersons().get(i).getClass().getSimpleName().equals("Borrower"))
                                {
                                    if(lib.getPersons().get(i).getID()==id)
                                    {
                                        set =false;
                                         bb=(Borrower)(lib.getPersons().get(i));
                                    }
                                }
                            }

                            set=true;
```

```
                        ArrayList<Loan> books = loans;

                        for(int i=0;i<books.size() && set;i++)
                        {
                            if(books.get(i).getBook().getID()==bid
&&books.get(i).getReceiver()==null )
                            {
                              set=false;
                              Loan                                          bBook=new
Loan(bb,books.get(i).getBook(),books.get(i).getIssuer(),null,books.get(i).getIssuedDate(),
null,books.get(i).getFineStatus());
                              bb.addBorrowedBook(bBook);
                            }
                        }

                    }while(rs.next());
            }

            ArrayList<Person> persons = lib.getPersons();

            /* Setting Person ID Count */
            int max=0;

            for(int i=0;i<persons.size();i++)
            {
                if (max < persons.get(i).getID())
                    max=persons.get(i).getID();
            }
            Person.setIDCount(max);
    }


    // Filling Changes back to Database
    public          void          fillItBack(Connection          con)          throws
SQLException,SQLIntegrityConstraintViolationException
    {
            /*-----------Loan Table Cleared------------*/

            String template ="DELETE FROM LIBRARY.LOAN";
            PreparedStatement stmts = con.prepareStatement(template);

            stmts.executeUpdate();

            /*-----------Borrowed Books Table Cleared------------*/

            template ="DELETE FROM LIBRARY.BORROWED_BOOK";
            stmts = con.prepareStatement(template);

            stmts.executeUpdate();

            /*-----------OnHoldBooks Table Cleared------------*/

            template ="DELETE FROM LIBRARY.ON_HOLD_BOOK";
```

```
            stmts = con.prepareStatement(template);

            stmts.executeUpdate();

            /*-----------Books Table Cleared-----------*/

            template ="DELETE FROM LIBRARY.BOOK";
            stmts = con.prepareStatement(template);

            stmts.executeUpdate();

            /*-----------Clerk Table Cleared-----------*/

            template ="DELETE FROM LIBRARY.CLERK";
            stmts = con.prepareStatement(template);

            stmts.executeUpdate();

            /*-----------Librarian Table Cleared-----------*/

            template ="DELETE FROM LIBRARY.LIBRARIAN";
            stmts = con.prepareStatement(template);

            stmts.executeUpdate();

            /*-----------Borrower Table Cleared-----------*/

            template ="DELETE FROM LIBRARY.BORROWER";
            stmts = con.prepareStatement(template);

            stmts.executeUpdate();

            /*-----------Staff Table Cleared-----------*/

            template ="DELETE FROM LIBRARY.STAFF";
            stmts = con.prepareStatement(template);

            stmts.executeUpdate();

            /*-----------Person Table Cleared-----------*/

            template ="DELETE FROM LIBRARY.PERSON";
            stmts = con.prepareStatement(template);

            stmts.executeUpdate();

            Library lib =this;

        /* Filling Person's Table*/
        for(int i=0;i<lib.getPersons().size();i++)
        {
            template ="INSERT INTO LIBRARY.PERSON
(ID,PNAME,PASSWORD,ADDRESS,PHONE_NO) values (?,?,?,?,?)";
            PreparedStatement stmt = con.prepareStatement(template);
```

```
            stmt.setInt(1, lib.getPersons().get(i).getID());
            stmt.setString(2, lib.getPersons().get(i).getName());
            stmt.setString(3,  lib.getPersons().get(i).getPassword());
            stmt.setString(4, lib.getPersons().get(i).getAddress());
            stmt.setInt(5, lib.getPersons().get(i).getPhoneNumber());

            stmt.executeUpdate();
        }

        /* Filling Clerk's Table and Staff Table*/
        for(int i=0;i<lib.getPersons().size();i++)
        {
            if (lib.getPersons().get(i).getClass().getSimpleName().equals("Clerk"))
            {
                template ="INSERT INTO LIBRARY.STAFF (S_ID,TYPE,SALARY) values
(?,?,?)";
                PreparedStatement stmt = con.prepareStatement(template);
                stmt.setInt(1,lib.getPersons().get(i).getID());
                stmt.setString(2,"Clerk");
                stmt.setDouble(3, ((Clerk)(lib.getPersons().get(i))).getSalary());
                stmt.executeUpdate();
                template ="INSERT INTO LIBRARY.CLERK (C_ID,DESK_NO) values
(?,?)";
                stmt = con.prepareStatement(template);
                stmt.setInt(1,lib.getPersons().get(i).getID());
                stmt.setInt(2, ((Clerk)(lib.getPersons().get(i))).deskNo);
                stmt.executeUpdate();
            }

        }

        if(lib.getLibrarian()!=null)      // if  librarian is there
            {
            template ="INSERT INTO LIBRARY.STAFF (S_ID,TYPE,SALARY) values
(?,?,?)";
            PreparedStatement stmt = con.prepareStatement(template);

            stmt.setInt(1, lib.getLibrarian().getID());
            stmt.setString(2,"Librarian");
            stmt.setDouble(3,lib.getLibrarian().getSalary());

            stmt.executeUpdate();

            template ="INSERT INTO LIBRARY.LIBRARIAN (L_ID,OFFICE_NO) values
(?,?)";
            stmt = con.prepareStatement(template);

            stmt.setInt(1,lib.getLibrarian().getID());
            stmt.setInt(2, lib.getLibrarian().officeNo);

            stmt.executeUpdate();
            }
```

```
        /* Filling Borrower's Table*/
        for(int i=0;i<lib.getPersons().size();i++)
        {
            if (lib.getPersons().get(i).getClass().getSimpleName().equals("Borrower"))
            {
                template ="INSERT INTO LIBRARY.BORROWER(B_ID) values (?)";
                PreparedStatement stmt = con.prepareStatement(template);
                stmt.setInt(1, lib.getPersons().get(i).getID());
                stmt.executeUpdate();
            }
        }

        ArrayList<Book> books = lib.getBooks();

        /*Filling Book's Table*/
        for(int i=0;i<books.size();i++)
        {
            template ="INSERT INTO LIBRARY.BOOK
(ID,TITLE,AUTHOR,SUBJECT,IS_ISSUED) values (?,?,?,?,?)";
            PreparedStatement stmt = con.prepareStatement(template);

            stmt.setInt(1,books.get(i).getID());
            stmt.setString(2,books.get(i).getTitle());
            stmt.setString(3, books.get(i).getAuthor());
            stmt.setString(4, books.get(i).getSubject());
            stmt.setBoolean(5, books.get(i).getIssuedStatus());
            stmt.executeUpdate();

        }

        /* Filling Loan Book's Table*/
        for(int i=0;i<loans.size();i++)
        {
            template ="INSERT INTO
LIBRARY.LOAN(L_ID,BORROWER,BOOK,ISSUER,ISS_DATE,RECEIVER,RET_DATE,FINE_PAI
D) values (?,?,?,?,?,?,?,?)";
            PreparedStatement stmt = con.prepareStatement(template);

            stmt.setInt(1,i+1);
            stmt.setInt(2,loans.get(i).getBorrower().getID());
            stmt.setInt(3,loans.get(i).getBook().getID());
            stmt.setInt(4,loans.get(i).getIssuer().getID());
            stmt.setTimestamp(5, new
java.sql.Timestamp(loans.get(i).getIssuedDate().getTime()));
            stmt.setBoolean(8,loans.get(i).getFineStatus());
            if(loans.get(i).getReceiver()==null)
            {
                stmt.setNull(6,Types.INTEGER);
                stmt.setDate(7,null);
            }
            else
            {
                stmt.setInt(6,loans.get(i).getReceiver().getID());
                stmt.setTimestamp(7, new
```

```
java.sql.Timestamp(loans.get(i).getReturnDate().getTime()));
                }

                stmt.executeUpdate();

            }

            /* Filling On_Hold_ Table*/

            int x=1;
            for(int i=0;i<lib.getBooks().size();i++)
            {
                for(int j=0;j<lib.getBooks().get(i).getHoldRequests().size();j++)
                {
                template ="INSERT INTO
LIBRARY.ON_HOLD_BOOK(REQ_ID,BOOK,BORROWER,REQ_DATE) values (?,?,?,?)";
                PreparedStatement stmt = con.prepareStatement(template);

                stmt.setInt(1,x);

stmt.setInt(3,lib.getBooks().get(i).getHoldRequests().get(j).getBorrower().getID());

stmt.setInt(2,lib.getBooks().get(i).getHoldRequests().get(j).getBook().getID());
                stmt.setDate(4,new
java.sql.Date(lib.getBooks().get(i).getHoldRequests().get(j).getRequestDate().getTime()));

                stmt.executeUpdate();
                x++;

                }
            }
            /*for(int i=0;i<lib.getBooks().size();i++)
            {
                for(int j=0;j<lib.getBooks().get(i).getHoldRequests().size();j++)
                {
                template = "INSERT INTO
LIBRARY.ON_HOLD_BOOK(REQ_ID,BOOK,BORROWER,REQ_DATE) values (?,?,?,?)";
                PreparedStatement stmt = con.prepareStatement(template);

                stmt.setInt(1,i+1);

stmt.setInt(3,lib.getBooks().get(i).getHoldRequests().get(j).getBorrower().getID());

stmt.setInt(2,lib.getBooks().get(i).getHoldRequests().get(j).getBook().getID());
                stmt.setDate(4,new
java.sql.Date(lib.getBooks().get(i).getHoldRequests().get(j).getRequestDate().getTime()));

                stmt.executeUpdate();
                }
            }*/

            /* Filling Borrowed Book Table*/
            for(int i=0;i<lib.getBooks().size();i++)
              {
```

```
                if(lib.getBooks().get(i).getIssuedStatus()==true)
                {
                    boolean set=true;
                    for(int j=0;j<loans.size() && set ;j++)
                    {
                        if(lib.getBooks().get(i).getID()==loans.get(j).getBook().getID())
                        {
                            if(loans.get(j).getReceiver()==null)
                            {
                              template ="INSERT INTO
LIBRARY.BORROWED_BOOK(BOOK,BORROWER) values (?,?)";
                              PreparedStatement stmt = con.prepareStatement(template);
                              stmt.setInt(1,loans.get(j).getBook().getID());
                              stmt.setInt(2,loans.get(j).getBorrower().getID());

                              stmt.executeUpdate();
                              set=false;
                            }
                        }

                    }

                }
            }
    } // Filling Done!




}   // Library Class Closed
```

//

[Loan.java]

```
package LMS;
import java.time.temporal.ChronoUnit;
import java.util.Date;
import java.util.Scanner;
public class Loan
{
    private Borrower borrower;
    private Book book;

    private Staff issuer;
    private Date issuedDate;

    private Date dateReturned;
    private Staff receiver;

    private boolean finePaid;

    public Loan(Borrower bor, Book b, Staff i, Staff r, Date iDate, Date rDate,
boolean fPaid)  // Para cons.
    {
        borrower = bor;
        book = b;
        issuer = i;
        receiver = r;
        issuedDate = iDate;
        dateReturned = rDate;

        finePaid = fPaid;
    }

    /*----- Getter FUNCs.----------*/

    public Book getBook()         //Returns the book
    {
        return book;
    }

    public Staff getIssuer()      //Returns the Staff Member who issued the book
    {
        return issuer;
    }

    public Staff getReceiver()  //Returns the Staff Member to whom book is returned
    {
        return receiver;
    }

    public Date getIssuedDate()       //Returns the date on which this particular book
was issued
    {
        return issuedDate;
```

```
    }
    public Date getReturnDate()       //Returns the date on which this particular book was returned
    {
        return dateReturned;
    }

    public Borrower getBorrower()     //Returns the Borrower to whom the book was issued
    {
        return borrower;
    }

    public boolean getFineStatus()  // Returns status of fine
    {
        return finePaid;
    }
    /*-----------------------------------------------*/


    /*----------Setter FUNCs.---------------------*/
    public void setReturnedDate(Date dReturned)
    {
        dateReturned = dReturned;
    }

    public void setFineStatus(boolean fStatus)
    {
        finePaid = fStatus;
    }

    public void setReceiver(Staff r)
    {
        receiver = r;
    }
    /*-----------------------------------------------*/

    //Computes fine for a particular loan only
    public double computeFine1()
    {
        //-----------Computing Fine-----------
        double totalFine =0;

        if (!finePaid)
        {
            Date iDate = issuedDate;
            Date rDate =new Date();
            long days =  ChronoUnit.DAYS.between(rDate.toInstant(), iDate.toInstant());

            days=0-days;
            days = days - Library.getInstance().book_return_deadline;
            if(days>0)
                totalFine = days * Library.getInstance().per_day_fine;
            else
```

```
                totalFine=0;
        }
        return totalFine;
    }


    public void payFine()
    {
        //-----------Computing Fine-----------//

        double totalFine = computeFine1();

        if (totalFine >0)
        {
            System.out.println("₩nTotal Fine generated: Rs "+ totalFine);
            System.out.println("Do you want to pay? (y/n)");

            Scanner input =new Scanner(System.in);

            String choice = input.next();

            if(choice.equals("y") || choice.equals("Y"))
                finePaid =true;

            if(choice.equals("n") || choice.equals("N"))
                finePaid =false;
        }
        else
        {
            System.out.println("₩nNo fine is generated.");
            finePaid =true;
        }
    }
    // Extending issued Date
    public void renewIssuedBook(Date iDate)
    {
        issuedDate = iDate;

        System.out.println("₩nThe deadline of the book "+ getBook().getTitle() +" has
been extended.");
        System.out.println("Issued Book is successfully renewed!₩n");
    }

}   // Loan class Closed
```

//////////////////////////////////////

[Main.java]

```
package LMS;
// Including Header Files.
import java.io.*;
import java.util.*;
import java.sql.*;
public class Main
{
    // Clearing Required Area of Screen
    public static void clrscr()
    {
        for (int i =0; i <20; i++)
            System.out.println();
    }
    // Asking for Input as Choice
    public static int takeInput(int min, int max)
    {
        String choice;
        Scanner input =new Scanner(System.in);

        while(true)
        {
            System.out.println("₩nEnter Choice: ");
            choice = input.next();
            if((!choice.matches(".*[a-zA-Z]+.*")) && (Integer.parseInt(choice) > min
&& Integer.parseInt(choice) < max))
            {
                return Integer.parseInt(choice);
            }

            else
                System.out.println("₩nInvalid Input.");
        }

    }
    // Functionalities of all Persons
    public static void allFunctionalities(Person person, int choice) throws IOException
    {
        Library lib = Library.getInstance();

        Scanner scanner =new Scanner(System.in);
        int input =0;

        //Search Book
        if (choice ==1)
        {
            lib.searchForBooks();
        }

        //Do Hold Request
        else if (choice ==2)
        {
```

```
                ArrayList<Book> books = lib.searchForBooks();

                if (books !=null)
                {
                    input = takeInput(-1,books.size());

                    Book b = books.get(input);

                    if("Clerk".equals(person.getClass().getSimpleName())
||"Librarian".equals(person.getClass().getSimpleName()))
                    {
                        Borrower bor = lib.findBorrower();
                        if (bor !=null)
                            b.makeHoldRequest(bor);
                    }
                    else
                        b.makeHoldRequest((Borrower)person);
                }
            }

            //View borrower's personal information
            else if (choice ==3)
            {
                if("Clerk".equals(person.getClass().getSimpleName())
||"Librarian".equals(person.getClass().getSimpleName()))
                {
                    Borrower bor = lib.findBorrower();

                    if(bor!=null)
                        bor.printInfo();
                }
                else
                    person.printInfo();
            }

            //Compute Fine of a Borrower
            else if (choice ==4)
            {
                if("Clerk".equals(person.getClass().getSimpleName())
||"Librarian".equals(person.getClass().getSimpleName()))
                {
                    Borrower bor = lib.findBorrower();

                    if(bor!=null)
                    {
                        double totalFine = lib.computeFine2(bor);
                        System.out.println("\nYour Total Fine is : Rs "+ totalFine );

                    }
                }
                else
                {
                    double totalFine = lib.computeFine2((Borrower)person);
                    System.out.println("\nYour Total Fine is : Rs "+ totalFine );
```

```
            }
        }

        //Check hold request queue of a book
        else if (choice ==5)
        {
            ArrayList<Book> books = lib.searchForBooks();

            if (books !=null)
            {
                input = takeInput(-1,books.size());
                books.get(input).printHoldRequests();
            }
        }

        //Issue a Book
        else if (choice ==6)
        {
            ArrayList<Book> books = lib.searchForBooks();
            if (books !=null)
            {
                input = takeInput(-1,books.size());
                Book b = books.get(input);

                Borrower bor = lib.findBorrower();
                if(bor!=null)
                {
                    b.issueBook(bor, (Staff)person);
                }
            }
        }
        //Return a Book
        else if (choice ==7)
        {
            Borrower bor = lib.findBorrower();
            if(bor!=null)
            {
                bor.printBorrowedBooks();
                ArrayList<Loan> loans = bor.getBorrowedBooks();

                if (!loans.isEmpty())
                {
                    input = takeInput(-1,loans.size());
                    Loan l = loans.get(input);

                    l.getBook().returnBook(bor, l, (Staff)person);
                }
                else
                    System.out.println("₩nThis borrower "+ bor.getName() +" has no
book to return.");
            }
        }
        //Renew a Book
```

```
        else if (choice ==8)
        {
            Borrower bor = lib.findBorrower();
            if(bor!=null)
            {
                bor.printBorrowedBooks();
                ArrayList<Loan> loans = bor.getBorrowedBooks();

                if (!loans.isEmpty())
                {
                    input = takeInput(-1,loans.size());
                    loans.get(input).renewIssuedBook(new java.util.Date());
                }
                else
                    System.out.println("₩nThis borrower "+ bor.getName() +" has no
issued book which can be renewed.");
            }
        }
        //Add new Borrower
        else if (choice ==9)
        {
            lib.createPerson('b');
        }
        //Update Borrower's Personal Info
        else if (choice ==10)
        {
            Borrower bor = lib.findBorrower();

            if(bor !=null)
                bor.updateBorrowerInfo();
        }

        //Add new Book
        else if (choice ==11)
        {
            BufferedReader reader =new BufferedReader(new
InputStreamReader(System.in));
            System.out.println("₩nEnter Title:");
            String title = reader.readLine();
            System.out.println("₩nEnter Subject:");
            String subject = reader.readLine();
            System.out.println("₩nEnter Author:");
            String author = reader.readLine();

            lib.createBook(title, subject, author);
        }

        //Remove a Book
        else if (choice ==12)
        {
            ArrayList<Book> books = lib.searchForBooks();

            if (books !=null)
            {
```

```
                input = takeInput(-1,books.size());

                lib.removeBookfromLibrary(books.get(input));
            }
        }
        //Change a Book's Info
        else if (choice ==13)
        {
            ArrayList<Book> books = lib.searchForBooks();

            if (books!=null)
            {
                input = takeInput(-1,books.size());

                books.get(input).changeBookInfo();
            }
        }

        //View clerk's personal information
        else if (choice ==14)
        {
            Clerk clerk = lib.findClerk();
            if(clerk!=null)
                clerk.printInfo();
        }

        // Functionality Performed.
        System.out.println("₩nPress any key to continue..₩n");
        scanner.next();
    }
```

```
/*-------------------------------------MAIN-------------------------
-------------------------*/

    public static void main(String[] args)
    {
        Scanner admin =new Scanner(System.in);

        //-------------------INTERFACE---------------------------//

        Library lib = Library.getInstance();

        // Setting some by default information like name of library ,fine, deadline and
limit of hold request
        lib.setFine(20);
        lib.setRequestExpiry(7);
        lib.setReturnDeadline(5);
        lib.setName("FAST Library");
```

```
        // Making connection with Database.
        Connection con = lib.makeConnection();

        if (con ==null)     // Oops can't connnect !
        {
            System.out.println("₩nError connecting to Database. Exiting.");
            return;
        }

        try {
        lib.populateLibrary(con);    // Populating Library with all Records

        boolean stop =false;
        while(!stop)
        {
            clrscr();
            // FRONT END //

System.out.println("-----------------------------------------------------
---");
            System.out.println("₩tWelcome to Library Management System");

System.out.println("-----------------------------------------------------
---");

            System.out.println("Following Functionalities are available: ₩n");
            System.out.println("1- Login");
            System.out.println("2- Exit");
            System.out.println("3- Admininstrative Functions"); // Administration has
access only


System.out.println("----------------------------------------₩n");

            int choice =0;
            choice = takeInput(0,4);

            if (choice ==3)
            {
                System.out.println("₩nEnter Password: ");
                String aPass = admin.next();

                if(aPass.equals("lib"))
                {
                    while (true)     // Way to Admin Portal
                    {
                        clrscr();

System.out.println("-----------------------------------------------------
---");
                            System.out.println("₩tWelcome to Admin's Portal");

System.out.println("-----------------------------------------------------
```

```
---");
                        System.out.println("Following Functionalities are available:
₩n");
                        System.out.println("1- Add Clerk");
                        System.out.println("2- Add Librarian");
                        System.out.println("3- View Issued Books History");
                        System.out.println("4- View All Books in Library");
                        System.out.println("5- Logout");

System.out.println("-----------------------------------------");
                        choice = takeInput(0,6);
                        if (choice ==5)
                            break;
                        if (choice ==1)
                            lib.createPerson('c');
                        else if (choice ==2)
                            lib.createPerson('l');
                        else if (choice ==3)
                            lib.viewHistory();
                        else if (choice ==4)
                            lib.viewAllBooks();

                        System.out.println("₩nPress any key to continue..₩n");
                        admin.next();
                    }
                }
                else
                    System.out.println("₩nSorry! Wrong Password.");
            }
            else if (choice ==1)
            {
                Person person = lib.login();
                if (person ==null){}

                else if (person.getClass().getSimpleName().equals("Borrower"))
                {
                    while (true)     // Way to Borrower's Portal
                    {
                        clrscr();


System.out.println("-------------------------------------------------------
---");
                        System.out.println("₩tWelcome to Borrower's Portal");

System.out.println("-------------------------------------------------------
---");
                        System.out.println("Following Functionalities are available:
₩n");
                        System.out.println("1- Search a Book");
                        System.out.println("2- Place a Book on hold");
                        System.out.println("3- Check Personal Info of Borrower");
                        System.out.println("4- Check Total Fine of Borrower");
                        System.out.println("5- Check Hold Requests Queue of a
```

```
Book");
                        System.out.println("6- Logout");

System.out.println("-----------------------------------------------------
---");

                        choice = takeInput(0,7);
                        if (choice ==6)
                            break;

                        allFunctionalities(person,choice);
                    }
                }

                else if (person.getClass().getSimpleName().equals("Clerk"))
                {
                    while(true) // Way to Clerk's Portal
                    {
                        clrscr();


System.out.println("-----------------------------------------------------
---");
                        System.out.println("₩tWelcome to Clerk's Portal");

System.out.println("-----------------------------------------------------
---");
                        System.out.println("Following Functionalities are available:
₩n");
                        System.out.println("1- Search a Book");
                        System.out.println("2- Place a Book on hold");
                        System.out.println("3- Check Personal Info of Borrower");
                        System.out.println("4- Check Total Fine of Borrower");

                        System.out.println("5- Check Hold Requests Queue of a
Book");
                        System.out.println("6- Check out a Book");
                        System.out.println("7- Check in a Book");

                        System.out.println("8- Renew a Book");
                        System.out.println("9- Add a new Borrower");
                        System.out.println("10- Update a Borrower's Info");
                        System.out.println("11- Logout");

System.out.println("-----------------------------------------------------
---");

                        choice = takeInput(0,12);
                        if (choice ==11)
                            break;

                        allFunctionalities(person,choice);
                    }
                }
```

```
                else if (person.getClass().getSimpleName().equals("Librarian"))
                {
                    while(true) // Way to Librarian Portal
                    {
                        clrscr();


System.out.println("------------------------------------------------------
---");
                        System.out.println("₩tWelcome to Librarian's Portal");

System.out.println("------------------------------------------------------
---");
                        System.out.println("Following Functionalities are available:
₩n");
                        System.out.println("1- Search a Book");
                        System.out.println("2- Place a Book on hold");
                        System.out.println("3- Check Personal Info of Borrower");
                        System.out.println("4- Check Total Fine of Borrower");
                        System.out.println("5- Check Hold Requests Queue of a
Book");
                        System.out.println("6- Check out a Book");
                        System.out.println("7- Check in a Book");

                        System.out.println("8- Renew a Book");
                        System.out.println("9- Add a new Borrower");
                        System.out.println("10- Update a Borrower's Info");
                        System.out.println("11- Add new Book");
                        System.out.println("12- Remove a Book");
                        System.out.println("13- Change a Book's Info");
                        System.out.println("14- Check Personal Info of Clerk");

                        System.out.println("15- Logout");

System.out.println("------------------------------------------------------
---");

                        choice = takeInput(0,16);
                        if (choice ==15)
                            break;

                        allFunctionalities(person,choice);
                    }
                }

            }
            else
                stop =true;
            System.out.println("₩nPress any key to continue..₩n");
            Scanner scanner =new Scanner(System.in);
            scanner.next();
        }
```

```
            //Loading back all the records in database
            lib.fillItBack(con);
            }
            catch(Exception e)
            {
                System.out.println("₩nExiting...₩n");
            }    // System Closed!

        }     // Main Closed

    }    // Class closed.
```

//

[Person.java]

```
package LMS;
public abstract class Person
{
    protected int id;              // ID of every person related to library
    protected String password;   // Password of every person related to library
    protected String name;        // Name of every person related to library
    protected String address;    // Address of every person related to library
    protected int phoneNo;        // PhoneNo of every person related to library

    static int currentIdNumber =0;       //This will be unique for every person, since it will be incremented when everytime
                                         //when a person is created
    public Person(int idNum, String name, String address, int phoneNum)     // para cons.
    {
        currentIdNumber++;

        if(idNum==-1)
        {
            id = currentIdNumber;
        }
        else
            id = idNum;

        password = Integer.toString(id);
        this.name = name;
        this.address = address;
        phoneNo = phoneNum;
    }

    // Printing Info of a Person
    public void printInfo()
    {
        System.out.println("-----------------------------------------");
        System.out.println("₩nThe details are: ₩n");
        System.out.println("ID: "+ id);
        System.out.println("Name: "+ name);
        System.out.println("Address: "+ address);
        System.out.println("Phone No: "+ phoneNo +"₩n");
    }

    /*---------Setter FUNCs.---------*/
    public void setAddress(String a)
    {
        address = a;
    }

    public void setPhone(int p)
    {
        phoneNo = p;
    }
```

```
    public void setName(String n)
    {
        name = n;
    }
    /*----------------------------*/

    /*-------Getter FUNCs.--------*/
    public String getName()
    {
        return name;
    }

    public String getPassword()
    {
        return password;
    }

     public String getAddress()
    {
        return address;
    }

     public int getPhoneNumber()
    {
        return phoneNo;
    }
    public int getID()
    {
        return id;
    }
    /*----------------------------*/

     public static void setIDCount(int n)
    {
        currentIdNumber=n;
    }

} // Person Class Closed
```

////////////////////////////////////

[Staff.java]

```
package LMS;
public class Staff extends Person
{
    protected double salary;

    public Staff(int id, String n, String a,int p, double s)
    {
        super(id,n,a,p);
        salary = s;
    }

    @Override
    public void printInfo()
    {
        super.printInfo();
        System.out.println("Salary: "+ salary +"₩n");
    }

    public double getSalary()
    {
        return salary;
    }
}
```