```
// Enums
enum BookType { SCI_FI, ROMANTIC, FANTASY, DRAMA; }
enum BookFormat { HARDCOVER, PAPERBACK, NEWSPAPER, JOURNAL; }
enum BookStatus { ISSUED, AVAILABLE, RESERVED, LOST; }
enum ReservationStatus { RESERVED, CANCELLED, COMPLETED; }

// Supporting classes
class Address {
    int pinCode;
    String street;
    String city;
    String state;
    String country;
    public Address(int pinCode, String street, String city, String state, String country) {
        this.pinCode = pinCode;
        this.street = street;
        this.city = city;
        this.state = state;
        this.country = country;
    }
}

class Rack {
    int number;
    String locationId;
    public Rack(int number, String locationId) {
        this.number = number;
        this.locationId = locationId;
    }
}

class Person {
    String firstName;
    String lastName;
    public Person(String firstName, String lastName) {
        this.firstName = firstName;
        this.lastName = lastName;
    }
}

class Author extends Person {
    List<Book> booksPublished;
    public Author(String firstName, String lastName) {
        super(firstName, lastName);
        this.booksPublished = new ArrayList<>();
    }
}

class Account {
    String userName;
    String password;
    int accountId;
    public Account(String userName, String password, int accountId) {
        this.userName = userName;
        this.password = password;
        this.accountId = accountId;
```

```
    }
}

// Book and BookItem
class Book {
    String uniqueIdNumber;
    String title;
    List<Author> authors;
    BookType bookType;
    public Book(String uniqueIdNumber, String title, List<Author> authors, BookType bookType) {
        this.uniqueIdNumber = uniqueIdNumber;
        this.title = title;
        this.authors = authors;
        this.bookType = bookType;
    }
}

class BookItem extends Book {
    String barcode;
    Date publicationDate;
    Rack rackLocation;
    BookStatus bookStatus;
    BookFormat bookFormat;
    Date issueDate;
    public BookItem(String uniqueIdNumber, String title, List<Author> authors, BookType bookType,
                    String barcode, Date publicationDate, Rack rackLocation, BookStatus bookStatus,
                    BookFormat bookFormat) {
        super(uniqueIdNumber, title, authors, bookType);
        this.barcode = barcode;
        this.publicationDate = publicationDate;
        this.rackLocation = rackLocation;
        this.bookStatus = bookStatus;
        this.bookFormat = bookFormat;
        this.issueDate = null;
    }
}

// Library
class Library {
    String name;
    Address location;
    List<BookItem> books;
    public Library(String name, Address location) {
        this.name = name;
        this.location = location;
        this.books = new ArrayList<>();
    }
    public void addBookItem(BookItem bookItem) {
        books.add(bookItem);
    }
    public boolean removeBookItem(String barcode) {
        return books.removeIf(book -> book.barcode.equals(barcode));
    }
    public List<BookItem> getBooks() {
        return books;
    }
}
```

```
// SystemUser, Member, Librarian
class SystemUser extends Person {
    String email;
    String phoneNumber;
    Account account;
    public SystemUser(String firstName, String lastName, String email, String phoneNumber, Account
account) {
        super(firstName, lastName);
        this.email = email;
        this.phoneNumber = phoneNumber;
        this.account = account;
    }
}

class Member extends SystemUser {
    int totalBookCheckedOut;
    Search searchObj;
    BookIssueService issueService;
    public Member(String firstName, String lastName, String email, String phoneNumber, Account account) {
        super(firstName, lastName, email, phoneNumber, account);
        this.totalBookCheckedOut = 0;
        this.searchObj = new Search();
        this.issueService = new BookIssueService();
    }
}

class Librarian extends SystemUser {
    Search searchObj;
    BookIssueService issueService;
    public Librarian(String firstName, String lastName, String email, String phoneNumber, Account account)
{
        super(firstName, lastName, email, phoneNumber, account);
        this.searchObj = new Search();
        this.issueService = new BookIssueService();
    }
    public void addBookItem(Library library, BookItem bookItem) {
        library.addBookItem(bookItem);
    }
    public boolean deleteBookItem(Library library, String barcode) {
        return library.removeBookItem(barcode);
    }
    public boolean editBookItem(Library library, BookItem newBookItem) {
        for (int i = 0; i < library.books.size(); i++) {
            if (library.books.get(i).barcode.equals(newBookItem.barcode)) {
                library.books.set(i, newBookItem);
                return true;
            }
        }
        return false;
    }
}

// BookLending, BookReservationDetail, BookIssueDetail
class BookLending {
    BookItem book;
    Date startDate;
```

```
    SystemUser user;
    public BookLending(BookItem book, Date startDate, SystemUser user) {
        this.book = book;
        this.startDate = startDate;
        this.user = user;
    }
}

class BookReservationDetail extends BookLending {
    ReservationStatus reservationStatus;
    public BookReservationDetail(BookItem book, Date startDate, SystemUser user, ReservationStatus
reservationStatus) {
        super(book, startDate, user);
        this.reservationStatus = reservationStatus;
    }
}

class BookIssueDetail extends BookLending {
    Date dueDate;
    public BookIssueDetail(BookItem book, Date startDate, SystemUser user, Date dueDate) {
        super(book, startDate, user);
        this.dueDate = dueDate;
    }
}

// Fine calculation class
class Fine {
    Date fineDate;
    BookItem book;
    SystemUser user;
    double fineAmount;

    public Fine(Date fineDate, BookItem book, SystemUser user, double fineAmount) {
        this.fineDate = fineDate;
        this.book = book;
        this.user = user;
        this.fineAmount = fineAmount;
    }

    // Fine policy: first 5 days ₹2/day, next 5 days ₹5/day, above 10 days ₹10/day [7]
    public static double calculateFine(int daysLate) {
        double fine = 0.0;
        if (daysLate <= 0) return 0.0;
        if (daysLate <= 5)
            fine = daysLate * 2.0;
        else if (daysLate <= 10)
            fine = 5 * 2.0 + (daysLate - 5) * 5.0;
        else
            fine = 5 * 2.0 + 5 * 5.0 + (daysLate - 10) * 10.0;
        return fine;
    }
}

// BookIssueService
class BookIssueService {
    Map<String, BookIssueDetail> issuedBooks = new HashMap<>();
    Map<String, BookReservationDetail> reservedBooks = new HashMap<>();
```

```
    Map<String, Fine> fines = new HashMap<>();

    public BookReservationDetail getReservationDetail(BookItem book) {
        return reservedBooks.get(book.barcode);
    }
    public void updateReservationDetail(BookReservationDetail detail) {
        reservedBooks.put(detail.book.barcode, detail);
    }
    public BookReservationDetail reserveBook(BookItem book, SystemUser user) {
        if (book.bookStatus == BookStatus.AVAILABLE) {
            BookReservationDetail detail = new BookReservationDetail(book, new Date(), user,
ReservationStatus.RESERVED);
            reservedBooks.put(book.barcode, detail);
            book.bookStatus = BookStatus.RESERVED;
            return detail;
        }
        return null;
    }
    public BookIssueDetail issueBook(BookItem book, SystemUser user) {
        if (book.bookStatus == BookStatus.AVAILABLE || book.bookStatus == BookStatus.RESERVED) {
            BookIssueDetail detail = new BookIssueDetail(book, new Date(), user, getDueDate());
            issuedBooks.put(book.barcode, detail);
            book.bookStatus = BookStatus.ISSUED;
            book.issueDate = new Date();
            reservedBooks.remove(book.barcode);
            return detail;
        }
        return null;
    }
    public BookIssueDetail renewBook(BookItem book, SystemUser user) {
        BookIssueDetail detail = issuedBooks.get(book.barcode);
        if (detail != null && detail.user.equals(user)) {
            detail.dueDate = getDueDate();
            return detail;
        }
        return null;
    }
    // Returns fine if any, else null
    public Fine returnBook(BookItem book, SystemUser user, Date actualReturnDate) {
        BookIssueDetail detail = issuedBooks.get(book.barcode);
        if (detail != null && detail.user.equals(user)) {
            int daysLate = getDaysLate(detail.dueDate, actualReturnDate);
            double fineAmount = Fine.calculateFine(daysLate);
            Fine fine = null;
            if (fineAmount > 0) {
                fine = new Fine(actualReturnDate, book, user, fineAmount);
                fines.put(book.barcode, fine);
            }
            issuedBooks.remove(book.barcode);
            book.bookStatus = BookStatus.AVAILABLE;
            book.issueDate = null;
            return fine;
        }
        return null;
    }
    private Date getDueDate() {
        Calendar cal = Calendar.getInstance();
```

```
        cal.add(Calendar.DAY_OF_MONTH, 14); // 2 weeks
        return cal.getTime();
    }
    private int getDaysLate(Date dueDate, Date actualReturnDate) {
        long diff = actualReturnDate.getTime() - dueDate.getTime();
        return (int) (diff / (1000 * 60 * 60 * 24));
    }
}

// Search
class Search {
    public List<BookItem> getBookByTitle(List<BookItem> books, String title) {
        List<BookItem> result = new ArrayList<>();
        for (BookItem book : books) {
            if (book.title.toLowerCase().contains(title.toLowerCase())) {
                result.add(book);
            }
        }
        return result;
    }
    public List<BookItem> getBookByAuthor(List<BookItem> books, Author author) {
        List<BookItem> result = new ArrayList<>();
        for (BookItem book : books) {
            if (book.authors.contains(author)) {
                result.add(book);
            }
        }
        return result;
    }
    public List<BookItem> getBookByType(List<BookItem> books, BookType bookType) {
        List<BookItem> result = new ArrayList<>();
        for (BookItem book : books) {
            if (book.bookType == bookType) {
                result.add(book);
            }
        }
        return result;
    }
    public List<BookItem> getBookByPublicationDate(List<BookItem> books, Date publicationDate) {
        List<BookItem> result = new ArrayList<>();
        for (BookItem book : books) {
            if (book.publicationDate.equals(publicationDate)) {
                result.add(book);
            }
        }
        return result;
    }
}
```

```
// Main class to demonstrate overdue fine handling
public class LibraryManagementSystemWithFine {
    public static void main(String[] args) {
        // Create library
        Address address = new Address(12345, "Main St", "City", "State", "Country");
        Library library = new Library("Central Library", address);

        // Create authors
        Author author1 = new Author("John", "Doe");
        List<Author> authors = new ArrayList<>();
        authors.add(author1);

        // Create books
        BookItem book1 = new BookItem("B001", "Java Programming", authors, BookType.SCI_FI, "BC001",
new Date(), new Rack(1, "A1"), BookStatus.AVAILABLE, BookFormat.HARDCOVER);

        // Add books to library
        library.addBookItem(book1);

        // Create users
        Account account1 = new Account("user1", "pass1", 1);
        Member member1 = new Member("Alice", "Brown", "alice@example.com", "1234567890", account1);

        // Member1 issues the book
        BookIssueDetail issueDetail = member1.issueService.issueBook(book1, member1);
        Date dueDate = issueDetail.dueDate;
        System.out.println("Member1 issued book: " + book1.title + ", Due date: " + dueDate);

        // Simulate late return (e.g., returned 8 days after due date)
        Calendar cal = Calendar.getInstance();
        cal.setTime(dueDate);
        cal.add(Calendar.DAY_OF_MONTH, 8); // 8 days late
        Date actualReturnDate = cal.getTime();

        Fine fine = member1.issueService.returnBook(book1, member1, actualReturnDate);
        if (fine != null) {
            System.out.println("Book returned late by 8 days. Fine to be paid: ₹" + fine.fineAmount);
        } else {
            System.out.println("Book returned on time. No fine.");
        }

        // Simulate on-time return
        issueDetail = member1.issueService.issueBook(book1, member1);
        dueDate = issueDetail.dueDate;
        cal.setTime(dueDate);
        Date onTimeReturn = cal.getTime();
        fine = member1.issueService.returnBook(book1, member1, onTimeReturn);
        if (fine != null) {
            System.out.println("Book returned late. Fine to be paid: ₹" + fine.fineAmount);
        } else {
            System.out.println("Book returned on time. No fine.");
        }
    }
}
```