ASSIGNMENT 03 (PBL)

Object Oriented Programming (CSC-210)

BS(IT) – 2A

Group Members

Name	Enrollment
1. MUHAMMAD HASAN	02-135251-040
2. SHAH RAIZ ASGHAR	02-135251-022
3. DANIYAL AFZAAL	02-135251-034

Submitted to:

Sir Abdullah Ayub

BAHRIA UNIVERSITY KARACHI CAMPUS

Department of Computer Science

Assignment 03 (PBL)

Question # 1 **(CLO1, PLO1, C2)** **Marks [05 x 1 = 05]**

Classify **the Object-Oriented Programming-enabled solution regarding Smart Library System.**

Your university wants to develop a Smart Library System that automates book borrowing, returning, and inventory tracking. The system should handle multiple types of users such as students, faculty, and librarians.

However, the task (s):

Design the class structure using inheritance and polymorphism. Ensure your design includes:

- Common properties and methods for all users.
- Role-specific functionalities (e.g., only librarians can add/remove books).
- A mechanism to track borrowing history.

Code:

```java
package Question01;

public class Book {
    private int id;
    private String title;
    private String author;
    private boolean availability;

    // constructor
    public Book(int id, String title, String author) {
        this.id = id;
        this.title = title;
        this.author = author;
        this.availability = true;
    }

    // getters
    public int getId() {
        return id;
    }
    public String getTitle() {
        return title;
    }
    public String getAuthor() {
        return author;
    }
    public boolean getAvailabilty() {
        return availability;
    }

    // setters
    public void setId(int id) {
        this.id = id;
    }
    public void setTitle(String title) {
        this.title = title;
    }
    public void setAuthor(String author) {
        this.author = author;
    }
    public void setAvailability(boolean availability) {
        this.availability = availability;
    }

}
```

```
package Question01;
import java.util.List;
import java.util.ArrayList;

public abstract class Member {
    private String id;
    private String name;
    private String email;
    protected List<Book> borrowedBooks; // protected because whenever a member
will borrow book, this attribute will be used
    protected int borrowingPeriod = 0;

    // constructpr
    public Member(String id, String name, String email) {
        this.id = id;
        this.name = name;
        this.email = email;
        borrowedBooks = new ArrayList<>(); // an empty array list of borrowed
books got created
    }
```

```java
    // getters
    public String getId() {
        return id;
    }
    public String getName() {
        return name;
    }
    public String getEmail() {
        return email;
    }
    public int getBorrowingPeriod() {
        return borrowingPeriod;
    }

    // setters
    public void setId(String id) {
        this.id = id;
    }
    public void setName(String name) {
        this.name = name;
    }
    public void setEmail(String email) {
        this.email = email;
    }
    // this is abstract, because every member will have different limit
    public abstract boolean setBorrowingPeriod(int borrowingPeriod);

    // method to display all borrowed books
    public void displayBorrowedBooks() {
        if (borrowedBooks == null || borrowedBooks.isEmpty()) {
            System.out.println("No books borrowed yet.");
            return;
        }
        System.out.println("\n----------- Borrowed Books -----------");
        System.out.println("Member: " + name + " (ID: " + id + ")");
        System.out.println("Borrowing Period: " + borrowingPeriod + " days");
        System.out.println("Total Books Borrowed: " + borrowedBooks.size());

        int i = 1;
        for (Book b : borrowedBooks) {
            System.out.println(i++ + ". " + b.getTitle() + " by " + b.getAuthor()
+ " (ID: " + b.getId() + ")");
        }
        System.out.println("------------------------------------------");
    }
    // as every member will have diff protocols to borrow a book
    public abstract boolean borrowBook(Book b);
    public abstract boolean returnBook(Book b);
}



package Question01;

public class Faculty extends Member{
    private final int MAX_BORROW_PERIOD = 10;
    private final int MAX_BORROW_BOOKS = 5;


    // constructor
    public Faculty(String id, String name, String email) {
        super(id, name, email);
    }
    // getters
    public int getMAX_BORROW_PERIOD() {
        return MAX_BORROW_PERIOD;
    }
    public int getMAX_BORROW_BOOKS() {
        return MAX_BORROW_BOOKS;
    }

    // setters
    @Override
    public boolean setBorrowingPeriod(int borrowingPeriod) {
        if(borrowingPeriod > MAX_BORROW_PERIOD){
            System.out.println("Borrowing period can't be that high.");
            return false;
        }

        this.borrowingPeriod = borrowingPeriod;
        return true;
    }
    @Override
    // here book will be passed after verfication that is it present in library
or not
    public boolean borrowBook(Book b) {
        if(borrowedBooks.size() >= MAX_BORROW_BOOKS){
            System.out.println("Max limit of borrowing books reached.");
            return false;
        }
        borrowedBooks.add(b);
        return true;
    }

    @Override
    public boolean returnBook(Book b) {

        // verifying that if user has actually borrowed that book
        for(int i = 0; i < borrowedBooks.size(); i++){
            if(borrowedBooks.get(i).getId() == b.getId()){
                return true;
            }
        }

        return false;
    }

}
```

```
package Question01;

public class Student extends Member{

    private final int MAX_BORROW_PERIOD = 7;
    private final int MAX_BORROW_BOOKS = 3;

    // constructor
    public Student(String id, String name, String email) {
        super(id, name, email);
    }

    // getters
    public int getMAX_BORROW_PERIOD() {
        return MAX_BORROW_PERIOD;
    }
    public int getMAX_BORROW_BOOKS() {
        return MAX_BORROW_BOOKS;
    }


    // setters
    @Override
    public boolean setBorrowingPeriod(int borrowingPeriod) {
        if(borrowingPeriod > MAX_BORROW_PERIOD){
            System.out.println("Borrowing period can't be that high.");
            return false;
        }

        this.borrowingPeriod = borrowingPeriod;
        return true;
    }

    @Override
    public boolean borrowBook(Book b) { // here book will be passed after
verfication that is it present in library or not
        if(borrowedBooks.size() >= MAX_BORROW_BOOKS){
            System.out.println("Max limit of borrowing books reached.");
            return false;
        }

        borrowedBooks.add(b);
        return true;
    }

    @Override
    public boolean returnBook(Book b) {

        // verifying that if user has actually borrowed that book
        for(int i = 0; i < borrowedBooks.size(); i++){
            if(borrowedBooks.get(i).getId() == b.getId()){
                return true;
            }
        }

        return false;
    }

}
```

```java
package Question01;
import java.util.*;

public class Library {
    private String name;
    private List<Book> books;
    private List<Member> members;

    // constructor
    public Library(String name){
        this.name = name;
        books = new ArrayList<>();
        members = new ArrayList<>();
    }

    // getter
    public String getName() {
        return name;
    }

    // setter
    public void setName(String name) {
        this.name = name;
    }

    // method to search book (helper method for finding book)
    public int searchBook(int id){
        for(int i = 0; i < books.size(); i++){
            if(books.get(i).getId() == id){
                return i; // returning the index
            }
        }
        return -1;
    }

    // find book
    public Book findBook(int id){
        int bookIndex = searchBook(id);

        // means no book in list with this id
        if(bookIndex == -1){
            return null;
        }
        return books.get(bookIndex);
    }

    // method to search member (helper method for finding member)
    public int searchMember(String id){
        for(int i = 0; i < members.size(); i++){
            if(members.get(i).getId().equals(id)){
                return i; // returning the index
            }
        }

        return -1;
    }

    // find member
    public Member findMember(String id){
        int memberIndex = searchMember(id);

        if(memberIndex == -1){
            return null;
        }

        return members.get(memberIndex);
    }


    /* all variables will be passed after verification in librarian */
    // method to add books
    public void addBook(Book b){
        books.add(b);
    }

    // method to remove books
    public void removeBook(Book b){
        int bookIndex = searchBook(b.getId());

        books.remove(bookIndex);
    }

    // method to add members
    public void addMember(Member m){
        members.add(m);
    }
}
```

```java
package Question01;

public class Librarian {
    private String id;
    private String name;
    private String email;
    private Library managedLibrary;

    // constructor
    public Librarian(String id, String name, String email, Library
managedLibrary) {
        this.id = id;
        this.name = name;
        this.email = email;
        this.managedLibrary = managedLibrary;
    }

    // getters
    public String getId() {
        return id;
    }
    public String getName() {
        return name;
    }
    public String getEmail() {
        return email;
    }

    // setters
    public void setId(String id) {
        this.id = id;
    }
    public void setName(String name) {
        this.name = name;
    }
    public void setEmail(String email) {
        this.email = email;
    }

    // method to add book
    public void addBook(Book b) {
        managedLibrary.addBook(b);
    }

    // method to add member
    public void addMember(Member m){
        managedLibrary.addMember(m);
    }

    // method to remove book
    public void removeBook(int id) {
        Book bookToRemove = managedLibrary.findBook(id);

        if (bookToRemove == null) {
            System.out.println("Book doesn't exist in library!");
            return;
        }

        managedLibrary.removeBook(bookToRemove);
        System.out.println("Book removed successfully from library!");
    }

    // method to issue book
    public void issueBook(String memberId, int bookId, int borrowTime) {

        // validating member
        Member member = managedLibrary.findMember(memberId);
        if (member == null) {
            System.out.println("Member doesn't exist.");
            return;
        }
        // validating book and it's availabilty
        Book book = managedLibrary.findBook(bookId);
        if (book == null || !(book.getAvailabilty())) {
            if(book == null){
                System.out.println("Book doesn't exist in library.");
            }
            else{
                System.out.println("Book is already borrowed by someone else.");
            }
            return;
        }

        /* now book and member is validated, book is also available. So issuing
book */

        // validating borrowing period
        if(!(member.setBorrowingPeriod(borrowTime))){
            return;
        }
        if (member.borrowBook(book)) { // true means book borrowed
            System.out.println("Book borrowed successfully!");
            book.setAvailability(false);
        }
        else{
            System.out.println("Not able to issue book.");
        }


        /*
         * Member member = managedLibrary.getMemberList().get(memberIndex);
         * Book book = managedLibrary.getBookList().get(bookIndex);
         *
         * Problems:
         *
         * You're accessing internal lists of Library through getters
         * This is called "breaking encapsulation" or "Law of Demeter violation"
         * Main shouldn't know Library has a "memberList" or "bookList"
         * Makes code fragile - if Library changes internal structure, this
breaks
         *
         * The rule: Don't chain more than 2 dots → object.method().method() is a
code
         * smell
         *
         * Law of demeter:
         * managedLibrary.getMemberList().get(index).borrowBook(book)
         *      ↓              ↓              ↓             ↓
         *    object       stranger1      stranger2     final method
         *
         * Solution:
         * make proper methods to get member and book in library class
         */

    }

    // method to recive book
    public void reciveBook(String memberId, int bookId) {

        // validating member
        Member member = managedLibrary.findMember(memberId);
        if (member == null) {
            System.out.println("Member doesn't exists.");
            return;
        }
```

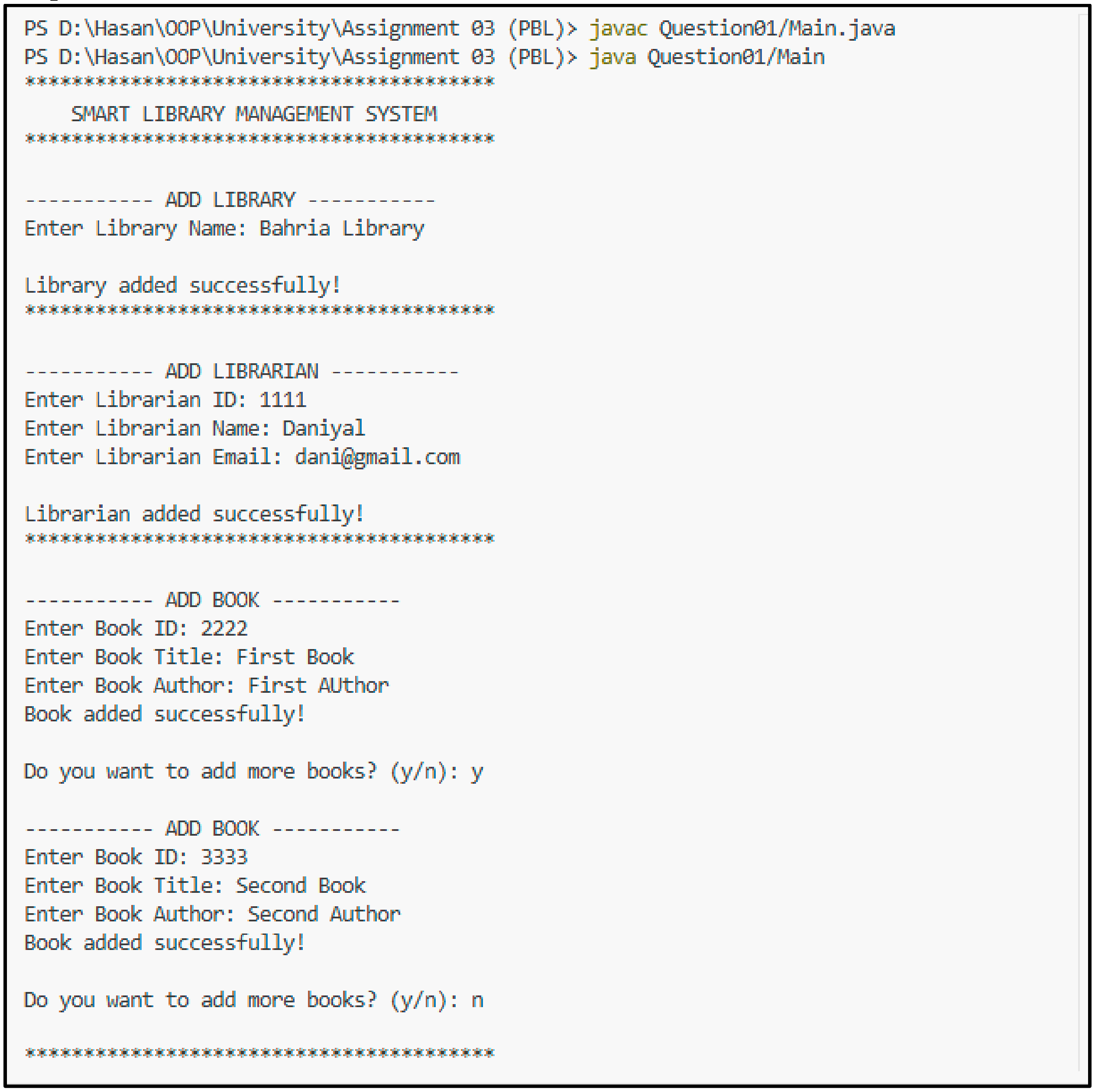
```java
        // validating book and it's availabilty (was it actually borrowed?)
        Book book = managedLibrary.findBook(bookId);
        if (book == null || book.getAvailabilty()) {
            if(book == null){
                System.out.println("Book doesn't exist in library.");
            }
            else{
                System.out.println("Book wasn't borrowed.");
            }
            return;
        }

        // now book and member is validated, book is also available. So reciving
book

        // setting borrowing period
        member.setBorrowingPeriod(0);

        if(member.returnBook(book)){
            System.out.println("Book recived successfully!");
            book.setAvailability(true);
        }
        else{
            System.out.println("Not able to recive book.");
        }

    }

    // method see borrwoing histroy
    public void seeBorrowingHistory(String id){

        // validating member
        Member member = managedLibrary.findMember(id);
        if (member == null) {
            System.out.println("Member doesn't exists.");
            return;
        }

        member.displayBorrowedBooks();
    }

}
```

```java
package Question01;
import java.util.Scanner;

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

        // Step 1: Add Librarian
        System.out.println("****************************************");
        System.out.println("    SMART LIBRARY MANAGEMENT SYSTEM    ");
        System.out.println("****************************************\n");

        System.out.println("----------- ADD LIBRARY -----------");
        System.out.print("Enter Library Name: ");
        String libraryName = sc.nextLine();

        Library library = new Library(libraryName);

        System.out.println("\nLibrary added successfully!");
        System.out.println("****************************************\n");


        System.out.println("----------- ADD LIBRARIAN -----------");
        System.out.print("Enter Librarian ID: ");
        String libId = sc.nextLine();
        System.out.print("Enter Librarian Name: ");
        String libName = sc.nextLine();
        System.out.print("Enter Librarian Email: ");
        String libEmail = sc.nextLine();

        Librarian librarian = new Librarian(libId, libName, libEmail, library);

        System.out.println("\nLibrarian added successfully!");
        System.out.println("****************************************\n");

        // Step 2: Add Books
        String choice;
        do {
            System.out.println("----------- ADD BOOK -----------");
            System.out.print("Enter Book ID: ");
            int bookId = sc.nextInt();
            sc.nextLine(); // consume newline

            System.out.print("Enter Book Title: ");
            String bookTitle = sc.nextLine();
            System.out.print("Enter Book Author: ");
            String bookAuthor = sc.nextLine();

            Book book = new Book(bookId, bookTitle, bookAuthor);
            librarian.addBook(book);

            System.out.println("Book added successfully!");

            System.out.print("\nDo you want to add more books? (y/n): ");
            choice = sc.nextLine();
            System.out.println();

        } while (choice.equalsIgnoreCase("y"));

        System.out.println("****************************************\n");

        // Step 3: Add Members
        do {
            System.out.println("----------- ADD MEMBER -----------");
            System.out.println("Select Member Type:");
            System.out.println("1. Student");
            System.out.println("2. Faculty");
            System.out.print("Enter choice: ");
            int memberType = sc.nextInt();
            sc.nextLine(); // consume newline

            System.out.print("Enter Member ID: ");
            String memberId = sc.nextLine();
            System.out.print("Enter Member Name: ");
            String memberName = sc.nextLine();
            System.out.print("Enter Member Email: ");
            String memberEmail = sc.nextLine();

            Member member;
            if (memberType == 1) {
                member = new Student(memberId, memberName, memberEmail);
            }
            else if (memberType == 2) {
                member = new Faculty(memberId, memberName, memberEmail);
            }
            else {
                System.out.println("Invalid choice!");
                System.out.println();
                continue;
            }

            librarian.addMember(member);
            System.out.println("Member added successfully!");

            System.out.print("\nDo you want to add more members? (y/n): ");
            choice = sc.nextLine();
            System.out.println();

        } while (choice.equalsIgnoreCase("y"));

        System.out.println("****************************************\n");

        // Main Menu
        int menuChoice;
        do {
            System.out.println("\n========== LIBRARY MENU ==========");
            System.out.println("1. Add member");
            System.out.println("2. Add book");
            System.out.println("3. Remove book");
            System.out.println("4. Issue book");
            System.out.println("5. Receive book");
            System.out.println("6. See borrowing history");
            System.out.println("7. Exit");
            System.out.println("==================================");
            System.out.print("Enter your choice: ");
            menuChoice = sc.nextInt();
            sc.nextLine(); // consume newline

            switch (menuChoice) {
                case 1: // Add member
                    System.out.println("\n----------- ADD MEMBER -----------");
                    System.out.println("Select Member Type:");
                    System.out.println("1. Student");
                    System.out.println("2. Faculty");
                    System.out.print("Enter choice: ");
                    int type = sc.nextInt();
                    sc.nextLine();

                    System.out.print("Enter Member ID: ");
                    String mId = sc.nextLine();
                    System.out.print("Enter Member Name: ");
                    String mName = sc.nextLine();
                    System.out.print("Enter Member Email: ");
                    String mEmail = sc.nextLine();
```

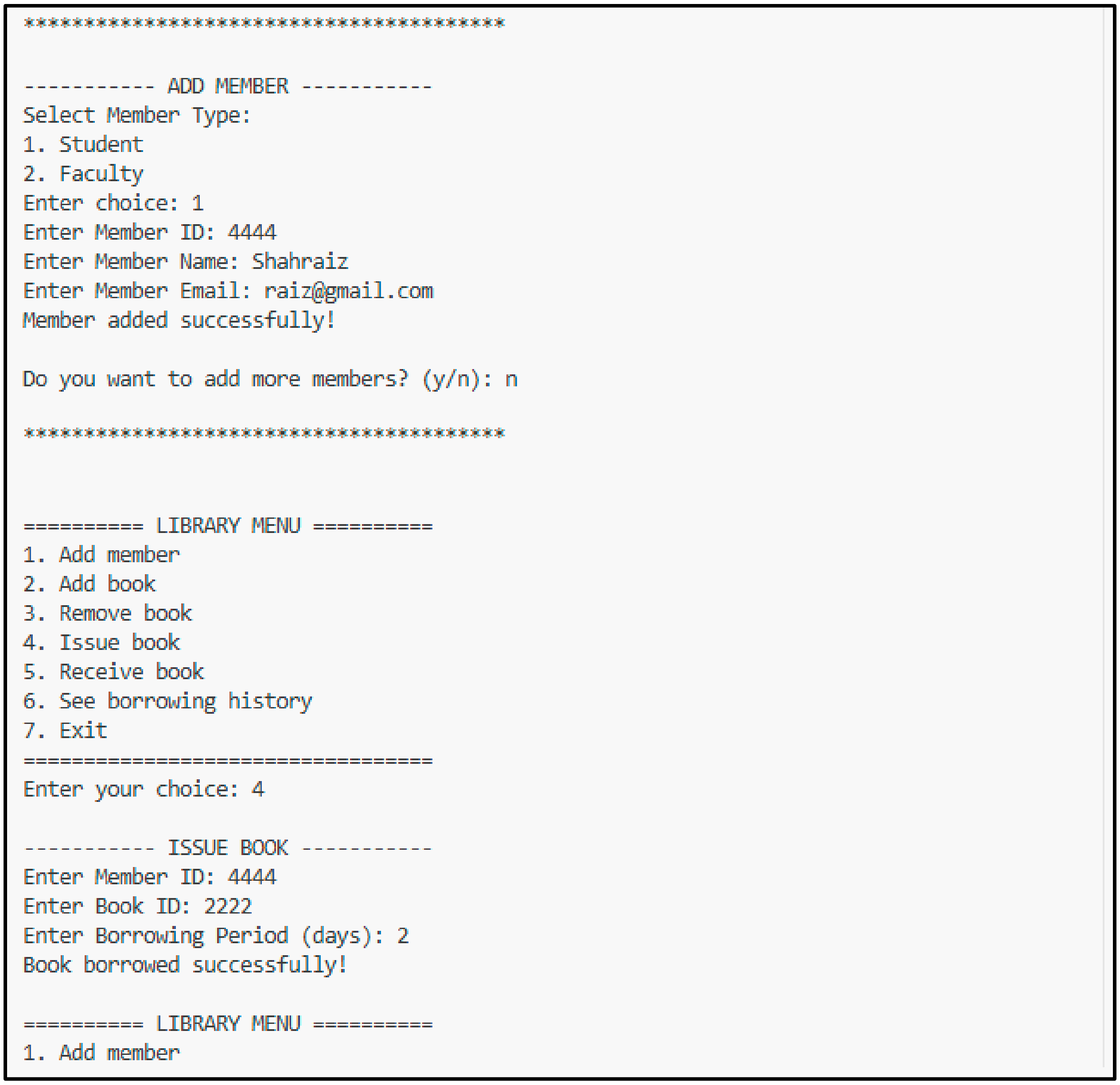
```java
            if (type == 1) {
                librarian.addMember(new Student(mId, mName, mEmail));
            }
            else if (type == 2) {
                librarian.addMember(new Student(mId, mName, mEmail));
            }
            else {
                System.out.println("Invalid choice!");
                break;
            }
            System.out.println("Member added successfully!");
            break;

        case 2: // Add book
            System.out.println("\n----------- ADD BOOK -----------");
            System.out.print("Enter Book ID: ");
            int bId = sc.nextInt();
            sc.nextLine();
            System.out.print("Enter Book Title: ");
            String bTitle = sc.nextLine();
            System.out.print("Enter Book Author: ");
            String bAuthor = sc.nextLine();

            librarian.addBook(new Book(bId, bTitle, bAuthor));
            System.out.println("Book added successfully!");
            break;

        case 3: // Remove book
            System.out.println("\n----------- REMOVE BOOK -----------");
            System.out.print("Enter Book ID to remove: ");
            int removeId = sc.nextInt();
            sc.nextLine();

            librarian.removeBook(removeId);
            break;

        case 4: // Issue book
            System.out.println("\n----------- ISSUE BOOK -----------");
            System.out.print("Enter Member ID: ");
            String issueMemberId = sc.nextLine();
            System.out.print("Enter Book ID: ");
            int issueBookId = sc.nextInt();
            System.out.print("Enter Borrowing Period (days): ");
            int borrowPeriod = sc.nextInt();
```

```
                    sc.nextLine();

                    librarian.issueBook(issueMemberId, issueBookId,
borrowPeriod);
                    break;

                case 5: // Receive book
                    System.out.println("\n----------- RECEIVE BOOK -----------");
                    System.out.print("Enter Member ID: ");
                    String returnMemberId = sc.nextLine();
                    System.out.print("Enter Book ID: ");
                    int returnBookId = sc.nextInt();
                    sc.nextLine();

                    librarian.reciveBook(returnMemberId, returnBookId);
                    break;

                case 6: // See borrowing history
                    System.out.println("\n----------- BORROWING HISTORY ---------
--");
                    System.out.print("Enter Member ID: ");
                    String historyMemberId = sc.nextLine();

                    librarian.seeBorrowingHistory(historyMemberId);
                    break;

                case 7: // Exit
                    System.out.println("\n*************************************
*");
                    System.out.println("Thank you for using the Library
System!");
                    System.out.println("*****************************************"
);
                    break;

                default:
                    System.out.println("\nInvalid choice!");
            }

        } while (menuChoice != 7);

        sc.close();
    }
}
```

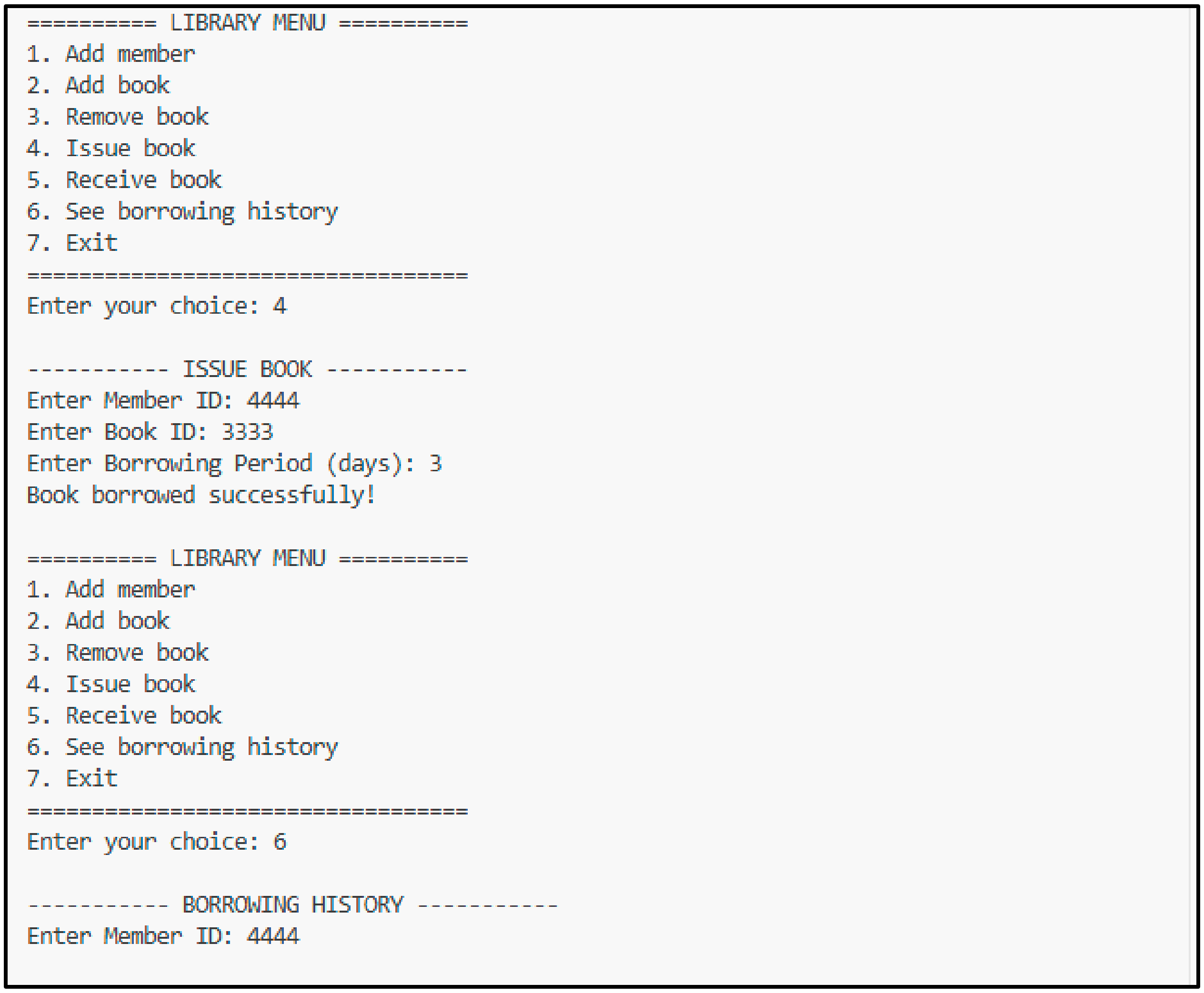
Output:

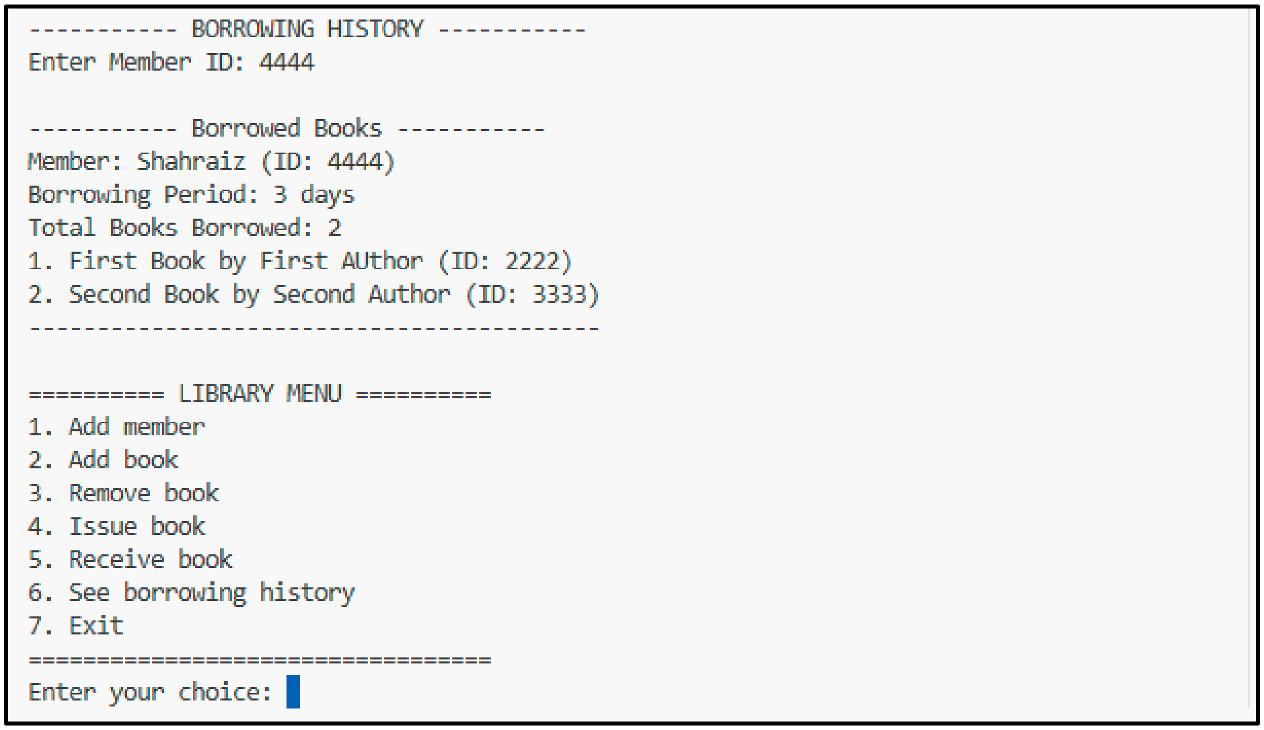


```
PS D:\Hasan\OOP\University\Assignment 03 (PBL)> javac Question01/Main.java
PS D:\Hasan\OOP\University\Assignment 03 (PBL)> java Question01/Main
****************************************
    SMART LIBRARY MANAGEMENT SYSTEM
****************************************


----------- ADD LIBRARY -----------
Enter Library Name: Bahria Library

Library added successfully!
****************************************


----------- ADD LIBRARIAN -----------
Enter Librarian ID: 1111
Enter Librarian Name: Daniyal
Enter Librarian Email: dani@gmail.com

Librarian added successfully!
****************************************


----------- ADD BOOK -----------
Enter Book ID: 2222
Enter Book Title: First Book
Enter Book Author: First AUthor
Book added successfully!

Do you want to add more books? (y/n): y

----------- ADD BOOK -----------
Enter Book ID: 3333
Enter Book Title: Second Book
Enter Book Author: Second Author
Book added successfully!

Do you want to add more books? (y/n): n

****************************************
```



```
*****************************************

----------- ADD MEMBER -----------
Select Member Type:
1. Student
2. Faculty
Enter choice: 1
Enter Member ID: 4444
Enter Member Name: Shahraiz
Enter Member Email: raiz@gmail.com
Member added successfully!

Do you want to add more members? (y/n): n

*****************************************



========== LIBRARY MENU ==========
1. Add member
2. Add book
3. Remove book
4. Issue book
5. Receive book
6. See borrowing history
7. Exit
==================================
Enter your choice: 4

----------- ISSUE BOOK -----------
Enter Member ID: 4444
Enter Book ID: 2222
Enter Borrowing Period (days): 2
Book borrowed successfully!

========== LIBRARY MENU ==========
1. Add member
```


```
========== LIBRARY MENU ==========
1. Add member
2. Add book
3. Remove book
4. Issue book
5. Receive book
6. See borrowing history
7. Exit
==================================
Enter your choice: 4

----------- ISSUE BOOK -----------
Enter Member ID: 4444
Enter Book ID: 3333
Enter Borrowing Period (days): 3
Book borrowed successfully!

========== LIBRARY MENU ==========
1. Add member
2. Add book
3. Remove book
4. Issue book
5. Receive book
6. See borrowing history
7. Exit
==================================
Enter your choice: 6

----------- BORROWING HISTORY -----------
Enter Member ID: 4444
```



```
----------- BORROWING HISTORY -----------
Enter Member ID: 4444

----------- Borrowed Books -----------
Member: Shahraiz (ID: 4444)
Borrowing Period: 3 days
Total Books Borrowed: 2
1. First Book by First AUthor (ID: 2222)
2. Second Book by Second Author (ID: 3333)
------------------------------------------

========== LIBRARY MENU ==========
1. Add member
2. Add book
3. Remove book
4. Issue book
5. Receive book
6. See borrowing history
7. Exit
==================================
Enter your choice:
```

Question # 2 **(CLO2, PLO2, C3)** **Marks [05 x 1 = 05]**

Apply the steps of program development and problem-solving techniques, especially in the problem of Ride-Sharing App Prototype.

Problem:

You are part of a startup developing a ride-sharing app like Uber. You need to model the interactions between drivers, riders, and trips.

Task:

Apply **encapsulation** and **abstraction** to design a class diagram that:

- Represents the main entities and their behaviors.
- Hides sensitive data (e.g., payment info, locations).

Allows scalability for features like rating drivers or multiple ride options.

Code:

```java
package Question02;

public class Vehicle {
    private String licensePlate;
    private String model;
    private String type;  // economy or premium

    public Vehicle(String licensePlate, String model, String type) {
        this.licensePlate = licensePlate;
        this.model = model;
        this.type = type;
    }

    public String getType() {
        return type;
    }
    public String getDetails() {
        return model + " (" + type + ") - " + licensePlate;
    }
}
```

```java
package Question02;

public abstract class User {
    protected String userId;
    protected String name;
    protected String phone;

    public User(String userId, String name, String phone) {
        this.userId = userId;
        this.name = name;
        this.phone = phone;
    }

    public abstract void displayProfile();
}
```

```java
package Question02;

public class Driver extends User {
    private Vehicle vehicle;
    private boolean available;

    public Driver(String userId, String name, String phone, Vehicle vehicle) {
        super(userId, name, phone);
        this.vehicle = vehicle;
        this.available = true;
    }

    public Vehicle getVehicle() {
        return vehicle;
    }
    public boolean isAvailable() {
        return available;
    }

    public void setAvailable(boolean available) {
        this.available = available;
    }

    @Override
    public void displayProfile() {
        System.out.println("Driver: " + name + ", Phone: " + phone + ", Vehicle: 
" + vehicle.getDetails());
    }
}
```

```java
package Question02;
import java.util.*;

public class Rider extends User {
    private PaymentInfo paymentInfo;
    private List<String> rideHistory = new ArrayList<>();

    public Rider(String userId, String name, String phone, PaymentInfo 
paymentInfo) {
        super(userId, name, phone);
        this.paymentInfo = paymentInfo;
    }

    public void addRideToHistory(String record) {
        rideHistory.add(record);
    }

    public List<String> getRideHistory() {
        return rideHistory;
    }

    @Override
    public void displayProfile() {
        System.out.println("Rider: " + name + ", Phone: " + phone + ", Payment: " 
+ paymentInfo.getMasked() + "Expiry: "
                + paymentInfo.getExpiry());
    }

    public PaymentInfo getPaymentInfo() {
        return paymentInfo;
    }
}
```

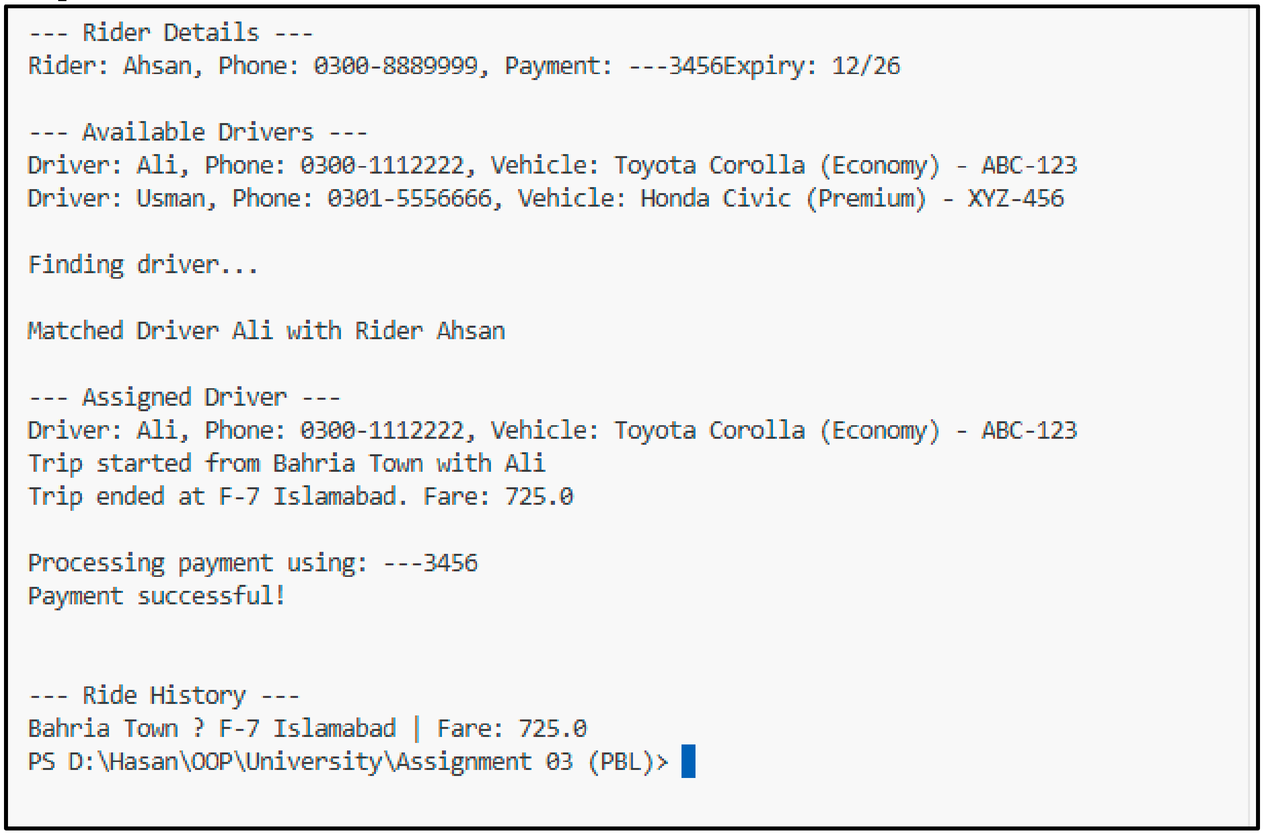
```
package Question02;
import java.util.*;

public class RideService {
    private List<Driver> drivers = new ArrayList<>();

    public void addDriver(Driver driver) {
        drivers.add(driver);
    }

    public Driver matchDriver(Rider rider) {
        for (Driver d : drivers) {
            if (d.isAvailable()) {
                d.setAvailable(false);
                System.out.println("Matched Driver " + d.name + " with Rider " +
rider.name);
                return d;
            }
        }
        System.out.println("No drivers available!");
        return null;
    }

    public void makeAvailable(Driver d) {
        d.setAvailable(true);
    }
}
```

```
package Question02;

public class PaymentInfo {
    private String maskedCard;
    private String expiry;

    public PaymentInfo(String cardNumber, String expiry) {
        this.maskedCard = mask(cardNumber);
        this.expiry = expiry;
    }
    private String mask(String card) {
        if (card == null || card.length() < 4) return "";
        return "---" + card.substring(card.length()-4);
    }
```

```java
    public String getMasked() { return maskedCard; }
    public String getExpiry() {return expiry;}

}



package Question02;

class Trip {
    private Rider rider;
    private Driver driver;
    private String start;
    private String end;
    private double distance;
    private double fare;

    public Trip(Rider rider, Driver driver, String start, String end, double
distance) {
        this.rider = rider;
        this.driver = driver;
        this.start = start;
        this.end = end;
        this.distance = distance;
        this.fare = calculateFare();
    }

    private double calculateFare() {
        double base = 100;
        double perKm = 50;
        return base + distance * perKm;
    }

    public void startTrip() {
        System.out.println("Trip started from " + start + " with " +
driver.name);
    }

    public void endTrip() {
        System.out.println("Trip ended at " + end + ". Fare: " + fare);
        rider.addRideToHistory(start + " → " + end + " | Fare: " + fare);
    }
}
```

```java
package Question02;

public class Main {
    public static void main(String[] args) {
        RideService rs = new RideService();

        // Create Drivers
        Driver d1 = new Driver("D01", "Ali", "0300-1112222", new Vehicle("ABC-
123", "Toyota Corolla", "Economy"));

        Driver d2 = new Driver("D02", "Usman", "0301-5556666", new Vehicle("XYZ-
456", "Honda Civic", "Premium"));

        rs.addDriver(d1);
        rs.addDriver(d2);

        // Create Rider
        Rider r1 = new Rider("R01", "Ahsan", "0300-8889999", new
PaymentInfo("1234567890123456", "12/26"));

        // Show rider details BEFORE booking
        System.out.println("\n--- Rider Details ---");
        r1.displayProfile();

        // Show all drivers
        System.out.println("\n--- Available Drivers ---");
        d1.displayProfile();
        d2.displayProfile();

        System.out.println("\nFinding driver...\n");
        Driver assigned = rs.matchDriver(r1);

        if (assigned == null)
            return;

        // Show assigned driver details
        System.out.println("\n--- Assigned Driver ---");
        assigned.displayProfile();

        // Start a trip
        Trip trip = new Trip(r1, assigned, "Bahria Town", "F-7 Islamabad", 12.5);

        trip.startTrip();
        trip.endTrip();
        // Simulate payment processing
        System.out.println("\nProcessing payment using: " +
r1.getPaymentInfo().getMasked());
        System.out.println("Payment successful!\n");

        // Make driver free again
        rs.makeAvailable(assigned);

        // Print Ride History
        System.out.println("\n--- Ride History ---");
        for (String h : r1.getRideHistory()) {
            System.out.println(h);
        }
    }
}
```

Output:



```
--- Rider Details ---
Rider: Ahsan, Phone: 0300-8889999, Payment: ---3456Expiry: 12/26

--- Available Drivers ---
Driver: Ali, Phone: 0300-1112222, Vehicle: Toyota Corolla (Economy) - ABC-123
Driver: Usman, Phone: 0301-5556666, Vehicle: Honda Civic (Premium) - XYZ-456

Finding driver...

Matched Driver Ali with Rider Ahsan

--- Assigned Driver ---
Driver: Ali, Phone: 0300-1112222, Vehicle: Toyota Corolla (Economy) - ABC-123
Trip started from Bahria Town with Ali
Trip ended at F-7 Islamabad. Fare: 725.0

Processing payment using: ---3456
Payment successful!


--- Ride History ---
Bahria Town ? F-7 Islamabad | Fare: 725.0
PS D:\Hasan\OOP\University\Assignment 03 (PBL)>
```

Question # 3 **(CLO3, PLO4, C4)** **Marks [05 x 1 = 05]**

Analyze the design and implementation of Online Examination System (OES) currently running throughout the globe.

Problem:

Due to a rise in online learning, your university wants to build an Online Examination System where instructors can create quizzes and students can take them online.

Task:

Use **OOP principles** to:

- Define the classes for Instructor, Student, Question, and Quiz.
- Include methods for adding questions, attempting quizzes, and grading.
- Discuss how **composition** and **aggregation** can be applied in your design.

Code:

```java
package Question03;

public class Question {
    private String question;

    // constructor
    public Question(String question) {
        this.question = question;
    }

    // getter and setter
    public String getQuestion() {
        return question;
    }
    public void setQuestion(String question) {
        this.question = question;
    }

    // method to display question
    public void displayQuestion(){
        System.out.println(question);
    }

}
```

```java
package Question03;
import java.util.*;

public class Quiz {
    private int quizNo;
    private List<Question> questionList;

    // constructor
    public Quiz(int quizNo){
        this.quizNo = quizNo;
        questionList = new ArrayList<>();
    }

    // getter
    public int getQuizNo() {
        return quizNo;
    }
    public List<Question> getQuestionList() {
        return questionList;
    }

    // setter
    public void setQuizNo(int quizNo) {
        this.quizNo = quizNo;
    }

    // method to add quiz
    public void addQuestion(Question q){
        questionList.add(q);
    }

    // method to display quiz
    public void displayQuiz(){

        int i = 1;

        System.out.println();
        for(Question q : questionList){
            System.out.print(i + ". ");
            q.displayQuestion();
            i++;
        }
    }

}
```

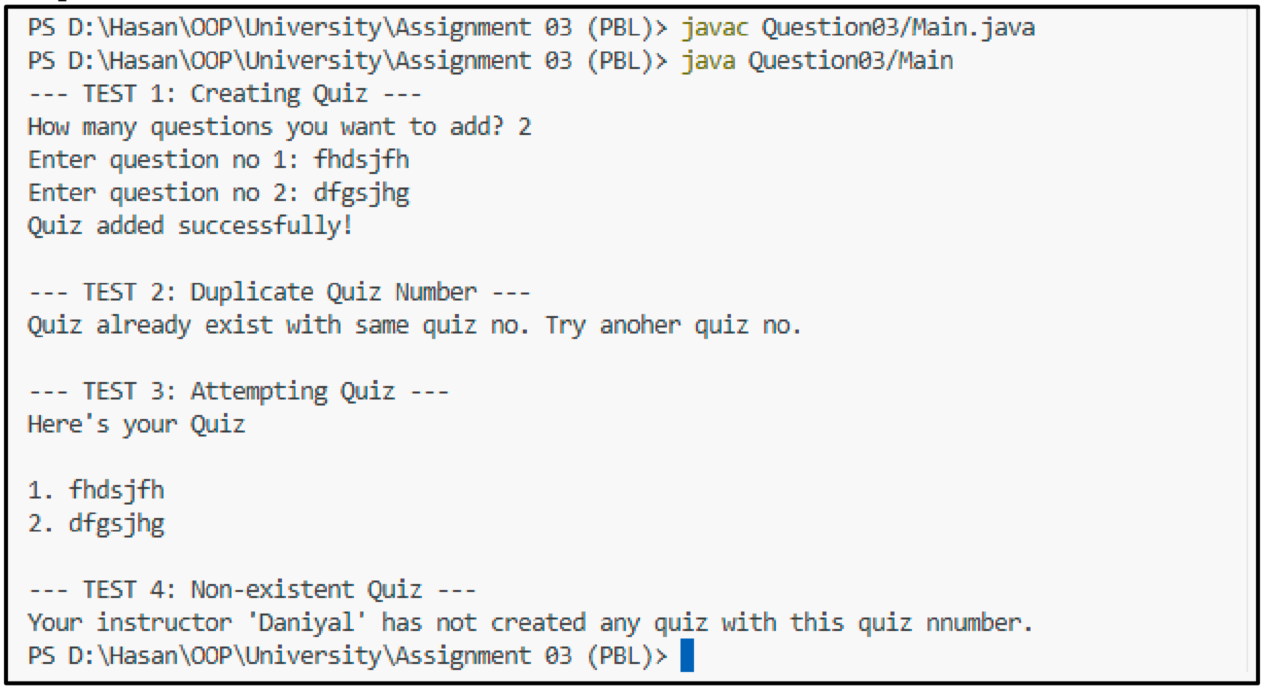
```java
package Question03;
import java.util.*;

public class Instructor {
    private int id;
    private String name;
    private String email;
    private List<Quiz> quizList;

    // constructor
    public Instructor(int id, String name, String email){
        this.id = id;
        this.name = name;
        this.email = email;
        quizList = new ArrayList<>();
    }

    // getters
    public int getId() {
        return id;
    }
    public String getName() {
        return name;
    }
    public String getEmail() {
        return email;
    }

    // setters
    public void setId(int id) {
        this.id = id;
    }
    public void setName(String name) {
        this.name = name;
    }
    public void setEmail(String email) {
        this.email = email;
    }

    // method to find quiz
    public Quiz findQuiz(int no){

        for(Quiz q : quizList){
            if(q.getQuizNo() == no){
                return q;
            }
        }

        return null;
    }

    // method to create quiz
    public void createQuiz(int no){

        // first verifying no quiz exists with the same number
        if(findQuiz(no) != null){
            System.out.println("Quiz already exist with same quiz no. Try anoher
quiz no.");
            return;
        }

        Scanner sc = new Scanner(System.in);

        // creating quiz
        Quiz quiz = new Quiz(no);

        System.out.print("How many questions you want to add? ");
        int noOfQUestions = sc.nextInt();
        sc.nextLine();

        for(int i = 0; i < noOfQUestions; i++){
            System.out.print("Enter question no " + (i+1) + ": ");
            String q = sc.nextLine();

            Question question = new Question(q);
            quiz.addQuestion(question);
        }

        quizList.add(quiz);
        System.out.println("Quiz added successfully!");

        // sc.close(); // closing scanner will not let you take input until the
program runs
    }

}
```

```
package Question03;

public class Student {
    private int id;
    private String name;
    private String email;
    private Instructor instructor;

    // constructor
    public Student(int id, String name, String email, Instructor instructor){
        this.id = id;
        this.name = name;
        this.email = email;
        this.instructor = instructor;
    }

    // getters
    public int getId(){
        return id;
    }
    public String getName(){
        return name;
    }
    public String getEmail(){
        return email;
    }
    public Instructor geInstructor(){
        return instructor;
    }

    // setters
    public void setId(int id){
        this.id = id;
    }
    public void setName(String name){
        this.name = name;
    }
    public void setEmail(String email){
        this.email = email;
    }
    public void setInstructor(Instructor instructor){
        this.instructor = instructor;
    }

    // method to attempt quiz
    public void attemptQuiz(int no){
        // first, check if the quiz with this(passed) quiz no present in the
student's instructor
        Quiz quizToAttempt = instructor.findQuiz(no);

        if(quizToAttempt == null){
            System.out.println("Your instructor '" + instructor.getName() + "'
has not created any quiz with this quiz nnumber.");
            return;
        }

        System.out.println("Here's your Quiz");
        quizToAttempt.displayQuiz();
    }
}
```

```
package Question03;

public class Main{

    public static void  main(String[] args) {
        // Setup instructors and students
        Instructor i1 = new Instructor(1111, "Hasan", "hasan@gmail.com");
        Instructor i3 = new Instructor(3333, "Daniyal", "daniyal@gmail.com");

        Student s1 = new Student(9999, "Arham", "arham@gmail.com", i1);
        Student s5 = new Student(5555, "Yasir", "yasir@gmail.com", i3);

        // Test 1: Create a quiz
        System.out.println("--- TEST 1: Creating Quiz ---");
        i1.createQuiz(10);

        // Test 2: Verify duplicate prevention
        System.out.println("\n--- TEST 2: Duplicate Quiz Number ---");
        i1.createQuiz(10);

        // Test 3: Student attempts quiz
        System.out.println("\n--- TEST 3: Attempting Quiz ---");
        s1.attemptQuiz(10);

        // Test 4: Quiz doesn't exist
        System.out.println("\n--- TEST 4: Non-existent Quiz ---");
```

```java
        s5.attemptQuiz(89);

    }
}
```

Output:



```
PS D:\Hasan\OOP\University\Assignment 03 (PBL)> javac Question03/Main.java
PS D:\Hasan\OOP\University\Assignment 03 (PBL)> java Question03/Main
--- TEST 1: Creating Quiz ---
How many questions you want to add? 2
Enter question no 1: fhdsjfh
Enter question no 2: dfgsjhg
Quiz added successfully!

--- TEST 2: Duplicate Quiz Number ---
Quiz already exist with same quiz no. Try anoher quiz no.

--- TEST 3: Attempting Quiz ---
Here's your Quiz

1. fhdsjfh
2. dfgsjhg

--- TEST 4: Non-existent Quiz ---
Your instructor 'Daniyal' has not created any quiz with this quiz nnumber.
PS D:\Hasan\OOP\University\Assignment 03 (PBL)>
```