INDEX

Week & Date	Idx	Program	Sign
4	4.1	**Smart Library Management System**	
	4.2	**Online Hotel Room Reservation System**	
	4.3	**Smart University Student Registration System**	

Problem 1

Problem: Smart Library Management System

Scenario Description

A college library has automated its book lending process using a **Smart Library Management System**. Every book in the library is represented as an object. Each book has its own details such as the book ID, title, author, price, and availability status.

The librarian should be able to:

- Add a new book to the library.
- Display the book details.
- Issue a book to a student.
- Return a book.
- Calculate the late fine if the book is returned after the due date.

Problem Statement

Create a Java program to manage a library book using a class named Book.

Book Details

Attribute	Data Type
bookId	int
title	String
author	String
price	double
available	boolean

Methods to Implement

1. addBook()

Stores the details of a book.

---

2. displayBook()

Displays all book details.

Sample Output

Book ID : 101
Title : Java Programming
Author : Herbert Scheldt
Price : ₹650.0
Available : Yes

---

3. issueBook()

If the book is available:

- Change the status to Not Available
- Display

Book issued successfully.
Otherwise display
Book is already issued.

---

4. returnBook(int lateDays)

When the book is returned:

- Change availability to Available
- Calculate the fine.

Rules
Fine = ₹10 × Late Days
If late days are 0,
Display
No Fine
Otherwise display
Fine Amount = ₹…

Create Object

In the main() method, Create one object

Book b1 = new Book();

Perform the following operations:

1. *Add Book*
2. *Display Book*
3. *Issue Book*
4. *Display Book*
5. *Return Book after 4 days*
6. *Display Book*

Java Solution

```
class Book {

    // Data Members
    int bookId;
    String title;
    String author;
    double price;
    boolean available;

    // Method to initialize book details
    void addBook(int id, String t, String a, double p) {

        bookId = id;
        title = t;
        author = a;
        price = p;
        available = true;
    }

    // Method to display details
    void displayBook() {

        System.out.println("\n------ Book Details ------");
        System.out.println("Book ID     : " + bookId);
        System.out.println("Title       : " + title);
        System.out.println("Author      : " + author);
```

```
        System.out.println("Price      : ₹" + price);

        if (available)
            System.out.println("Status     : Available");
        else
            System.out.println("Status     : Issued");
    }

    // Method to issue book
    void issueBook() {
        if (available) {
            available = false;
            System.out.println("\nBook issued successfully.");

        } else {
            System.out.println("\nBook is already issued.");
        }
    }

    // Method to return book
    void returnBook(int lateDays) {
        available = true;
        System.out.println("\nBook returned successfully.");
        if (lateDays == 0)
            System.out.println("No Fine");
        else
            System.out.println("Fine Amount : Rs." + (lateDays * 10));
    }
}


public class LibraryManagement {
    public static void main(String[] args) {
        Book b1 = new Book();
        b1.addBook(
                101,
                "Java Programming",
                "Herbert Schildt",
                650);

        b1.displayBook();
        b1.issueBook();
```

```
        b1.displayBook();
        b1.returnBook(4);
        b1.displayBook();
    }
}
```

Sample Output

```
D:\AY2026_27\JAVA\LAB\Week4>java LibraryManagement

------ Book Details ------
Book ID     : 101
Title       : Java Programming
Author      : Herbert Schildt
Price       : Rs.650.0
Status      : Available

Book issued successfully.

------ Book Details ------
Book ID     : 101
Title       : Java Programming
Author      : Herbert Schildt
Price       : Rs.650.0
Status      : Issued

Book returned successfully.
Fine Amount : â??40

------ Book Details ------
Book ID     : 101
Title       : Java Programming
Author      : Herbert Schildt
Price       : Rs.650.0
Status      : Available
```

Concepts Covered

Java Concept	Demonstrated By
Class	Book class
Object	Book b1 = new Book();
Data Members	bookId, title, author, price, available
Methods	addBook(), displayBook(), issueBook(), returnBook()

Java Concept	Demonstrated By
Method Parameters	lateDays
Method Invocation	b1.issueBook()
Object State	Book availability changes after issue/return
Decision Making	if-else statements inside methods

Outcomes

It naturally demonstrates the **object-oriented approach**:

- A **Class** (Book) acts as a blueprint.
- An **Object** (b1) represents a real book in the library.
- **Methods** define the actions that can be performed on a book.
- The **state of the object** (availability) changes when methods like issueBook() and returnBook() are called.

Problem 2

Problem: Online Hotel Room Reservation System

Scenario Description

A luxury hotel has introduced an **Online Hotel Reservation System** to simplify room bookings. Every time a guest books a room, a **Room object** is created automatically with the guest's booking details.

Since every room must be initialized with valid information **at the time of object creation**, the hotel software uses **constructors** instead of separate initialization methods.

Problem Statement

Create a Java program for a Hotel Room Reservation System using constructors.

Design a class named Room with the following data members.

Data Member	Data Type
roomNumber	int
guestName	String
roomType	String
numberOfDays	int
roomRentPerDay	double

Constructor Requirements

1. Default Constructor

When no booking information is available, initialize the object with default values.

Guest Name : Not Assigned
Room Type : Standard
Room Number : 0
Days : 0
Rent : ₹0

Display
Default Room Created Successfully.

2. Parameterized Constructor

Accept

- Room Number
- Guest Name
- Room Type
- Number of Days
- Rent Per Day

Initialize the object using the constructor.

Display

Room Booked Successfully.

---

3. Copy Constructor (User-Defined)

Create another room object by copying the details of an existing room.

Display

Room Details Copied Successfully.

Method

Create a method

void displayRoom()

to display

Room Number
Guest Name
Room Type
Number of Days
Rent Per Day
Total Rent

where

Total Rent = Number of Days × Rent Per Day

Create Objects

In the *main()* method,

Create three objects.

Object 1

Using the default constructor

Room room1 = new Room();

Object 2

Using the parameterized constructor

Room room2 = new Room(205, "Rahul Sharma", "Deluxe", 4, 3500);

Object 3

Using the copy constructor

Room room3 = new Room(room2);

Display the details of all three objects.

Java Solution

```
class Room {

    int roomNumber;
    String guestName;
    String roomType;
    int numberOfDays;
    double roomRentPerDay;
```

```
// Default Constructor
Room() {

    roomNumber = 0;
    guestName = "Not Assigned";
    roomType = "Standard";
    numberOfDays = 0;
    roomRentPerDay = 0;

    System.out.println("Default Room Created Successfully.");
}

// Parameterized Constructor
Room(int roomNumber, String guestName,
     String roomType, int numberOfDays,
     double roomRentPerDay) {

    this.roomNumber = roomNumber;
    this.guestName = guestName;
    this.roomType = roomType;
    this.numberOfDays = numberOfDays;
    this.roomRentPerDay = roomRentPerDay;

    System.out.println("Room Booked Successfully.");
}

// Copy Constructor
Room(Room r) {

    roomNumber = r.roomNumber;
    guestName = r.guestName;
    roomType = r.roomType;
    numberOfDays = r.numberOfDays;
    roomRentPerDay = r.roomRentPerDay;

    System.out.println("Room Details Copied Successfully.");
}

void displayRoom() {

    double totalRent = numberOfDays * roomRentPerDay;
```

```
        System.out.println("\n---------- Room Details ----------");
        System.out.println("Room Number      : " + roomNumber);
        System.out.println("Guest Name       : " + guestName);
        System.out.println("Room Type        : " + roomType);
        System.out.println("Number of Days   : " + numberOfDays);
        System.out.println("Rent Per Day     : ₹" + roomRentPerDay);
        System.out.println("Total Rent       : ₹" + totalRent);
    }

}

public class HotelReservation {

    public static void main(String[] args) {

        Room room1 = new Room();

        Room room2 = new Room(
                205,
                "Rahul Sharma",
                "Deluxe",
                4,
                3500);

        Room room3 = new Room(room2);

        room1.displayRoom();
        room2.displayRoom();
        room3.displayRoom();

    }
}
```

Sample Output

```
D:\AY2026_27\JAVA\LAB\Week4>javac HotelReservation.java

D:\AY2026_27\JAVA\LAB\Week4>java HotelReservation
Default Room Created Successfully.
Room Booked Successfully.
Room Details Copied Successfully.

---------- Room Details ----------
Room Number      : 0
Guest Name       : Not Assigned
Room Type        : Standard
Number of Days   : 0
Rent Per Day     : Rs.0.0
Total Rent       : Rs.0.0

---------- Room Details ----------
Room Number      : 205
Guest Name       : Rahul Sharma
Room Type        : Deluxe
Number of Days   : 4
Rent Per Day     : Rs.3500.0
Total Rent       : Rs.14000.0

---------- Room Details ----------
Room Number      : 205
Guest Name       : Rahul Sharma
Room Type        : Deluxe
Number of Days   : 4
Rent Per Day     : Rs.3500.0
Total Rent       : Rs.14000.0
```

Concepts Covered

Java Concept	Demonstrated By
Class	Room
Object	room1, room2, room3
Default Constructor	Room()
Parameterized Constructor	Room(int, String, String, int, double)
Copy Constructor (User-defined)	Room(Room r)
Constructor Overloading	Three constructors with different parameter lists
this Keyword	Used in the parameterized constructor
Method	displayRoom()
Object Initialization	Through constructors
Constructor Invocation	Automatic execution during object creation

---

OUTCOME

Students learn:

- **Default Constructor:** Creating a placeholder room when complete booking information is unavailable.
- **Parameterized Constructor:** Creating a fully initialized room reservation in a single statement.
- **Copy Constructor:** Duplicating an existing reservation, useful when creating similar bookings or backup records.
- **Constructor Overloading:** Using multiple constructors to initialize objects in different ways.

Problem 3

Problem: Smart University Student Registration System

Scenario Description

A university has developed a **Student Registration System** to manage admissions for the new academic year.

The university follows these rules:

- Every student has a **unique student ID**, **name**, and **department**.
- The **university name** is the same for every student and never changes.
- The **tuition fee** is fixed for all students during the academic year and cannot be modified.
- The university wants to keep track of the **total number of students registered**.

Instead of storing the university name and student count separately in every object, the software uses **static variables**. Since the tuition fee and university name should never change, they are declared as **final variables**.

This scenario demonstrates the use of both **static** and **final** variables in a practical application.

Problem Statement

Create a Java program to implement a **Student Registration System**.

Design a class named **Student** with the following data members.

Variable	Data Type	Modifier
studentId	int	Instance Variable
studentName	String	Instance Variable
department	String	Instance Variable
studentCount	int	static
UNIVERSITY_NAME	String	static final

Variable	Data Type	Modifier
TUITION_FEE	double	final

Requirements

1. static Variable
Maintain the total number of students admitted.
static int studentCount;
Whenever a new student object is created,
Increase the count by **1**.

---

2. static final Variable
The university name is common to all students and cannot be changed.
static final String UNIVERSITY_NAME = "ABC University";

---

3. final Variable
The tuition fee is fixed for each student.
final double TUITION_FEE = 75000;
The tuition fee **cannot be modified** after initialization.

---

4. Constructor
Accept
- Student ID
- Student Name
- Department

Initialize the object.
Increment
studentCount

---

5. Method
Create
void displayStudent ()
Display
- University Name
- Student ID
- Student Name
- Department
- Tuition Fee

---

6. Static Method
Create

static void displayStudentCount()
Display
Total Registered Students: ...

Create Objects

Create three students.

```
Student s1 = new Student(101, "Rahul", "CSE");
Student s2 = new Student(102, "Anjali", "ECE");
Student s3 = new Student(103, "Kiran", "AIML");
```

Display

- Details of each student.
- Total number of registered students.

Java Solution

```
class Student {

    int studentId;
    String studentName;
    String department;

    // Static variable (shared by all objects)
    static int studentCount = 0;

    // Static final variable (common constant)
    static final String UNIVERSITY_NAME = "LBR University";

    // Final instance variable
    final double TUITION_FEE = 75000;

    Student(int studentId, String studentName, String department) {

        this.studentId = studentId;
        this.studentName = studentName;
        this.department = department;

        studentCount++;
    }
```

```
    void displayStudent() {

        System.out.println("\n------ Student Details ------");
        System.out.println("University    : " + UNIVERSITY_NAME);
        System.out.println("Student ID    : " + studentId);
        System.out.println("Student Name  : " + studentName);
        System.out.println("Department    : " + department);
        System.out.println("Tuition Fee   : ₹" + TUITION_FEE);
    }

    static void displayStudentCount() {

        System.out.println("\nTotal Registered Students : " + studentCount);

    }

}

public class UniversityRegistration {

    public static void main(String[] args) {

        Student s1 = new Student(
                101,
                "Rahul",
                "CSE");

        Student s2 = new Student(
                102,
                "Anjali",
                "ECE");

        Student s3 = new Student(
                103,
                "Kiran",
                "AIML");

        s1.displayStudent();
        s2.displayStudent();
        s3.displayStudent();

        Student.displayStudentCount();

    }

}
```

Sample Output

```
D:\AY2026_27\JAVA\LAB\Week4>javac UniversityRegistration.java

D:\AY2026_27\JAVA\LAB\Week4>java UniversityRegistration

------ Student Details ------
University     : LBR University
Student ID     : 101
Student Name   : Rahul
Department     : CSE
Tuition Fee    : Rs.79000.0

------ Student Details ------
University     : LBR University
Student ID     : 102
Student Name   : Anjali
Department     : ECE
Tuition Fee    : Rs.79000.0

------ Student Details ------
University     : LBR University
Student ID     : 103
Student Name   : Kiran
Department     : AIML
Tuition Fee    : Rs.79000.0

Total Registered Students : 3
```

Java Concepts Covered

Java Concept	Demonstrated By
Class	Student
Object	s1, s2, s3
Instance Variables	studentId, studentName, department
static Variable	studentCount
final Variable	TUITION_FEE
static final Variable	UNIVERSITY_NAME

Java Concept	Demonstrated By
Constructor	Initializes student details and updates count
Instance Method	displayStudent()
Static Method	displayStudentCount()