Week 7 - S7 - Core OOP - Polymorphism - Assignment Problem(HW)

Name:Ramesh Harisabapathi Chettiar

Date of Submission:24/09/25

PROBLEM 1: Hotel Booking System

Concept: Method Overloading

You're building a hotel reservation system that calculates room prices in various ways:

● Standard booking (just room type and nights)

● Seasonal booking (room type, nights + seasonal multiplier)

● Corporate booking (room type, nights + corporate discount + meal package)

● Wedding package (room type, nights + guest count + decoration fee + catering options)

Each calculation should display a detailed breakdown of costs and savings applied.

Hint: Multiple ways to book the same room - let method signatures handle the complexity!

PROGRAM→

```java
/*
 * You're building a hotel reservation system that calculates room prices in various ways:
 * • Standard booking (just room type and nights)
 * • Seasonal booking (room type, nights + seasonal multiplier)
 * • Corporate booking (room type, nights + corporate discount + meal package)
 * • Wedding package (room type, nights + guest count + decoration fee + catering options)
 * Each calculation should display a detailed breakdown of costs and savings applied.
 * Hint: Multiple ways to book the same room - let method signatures handle the complexity!
 */

class Room {
    String type;
    double basePricePerNight;

    Room(String type, double basePricePerNight) {
        this.type = type;
        this.basePricePerNight = basePricePerNight;
    }
}

class BookingCalculator {
    // Standard booking
    static double calculateStandard(Room room, int nights) {
        return room.basePricePerNight * nights;
    }

    // Seasonal booking
    static double calculateSeasonal(Room room, int nights, double seasonalMultiplier) {
        return room.basePricePerNight * nights * seasonalMultiplier;
    }

    // Corporate booking
    static double calculateCorporate(Room room, int nights, double discountPercent, double mealPackagePerNight) {
        double base = room.basePricePerNight * nights;
        double discount = base * (discountPercent / 100.0);
        double meals = mealPackagePerNight * nights;
        return base - discount + meals;
    }

    // Wedding package
    static double calculateWedding(Room room, int nights, int guestCount, double decorationFee, double cateringPerGuest) {
        double base = room.basePricePerNight * nights;
        double catering = cateringPerGuest * guestCount;
        return base + decorationFee + catering;
    }
}

public class HotelReservationSystem {
    Run main | Debug main
    public static void main(String[] args) {
        Room deluxe = new Room("Deluxe", 3000);
        Room suite = new Room("Suite", 5000);

        // Standard booking
        int nights = 3;
        double stdTotal = BookingCalculator.calculateStandard(deluxe, nights);
        System.out.println("--- Standard Booking ---");
        System.out.println("Room: " + deluxe.type);
        System.out.println("Nights: " + nights);
        System.out.println("Base Price: Rs. " + deluxe.basePricePerNight + " x " + nights + " = Rs. " + stdTotal);
```

```
        // Seasonal booking
        double seasonalMultiplier = 1.2;
        double seasonalTotal = BookingCalculator.calculateSeasonal(suite, nights, seasonalMultiplier);
        System.out.println("\n--- Seasonal Booking ---");
        System.out.println("Room: " + suite.type);
        System.out.println("Nights: " + nights);
        System.out.println("Base Price: Rs. " + suite.basePricePerNight + " x " + nights + " = Rs. " + (suite.basePricePerNight * nights));
        System.out.println("Seasonal Multiplier: " + seasonalMultiplier);
        System.out.println("Total Price: Rs. " + seasonalTotal);

        // Corporate booking
        double discountPercent = 15;
        double mealPackagePerNight = 500;
        double corpTotal = BookingCalculator.calculateCorporate(deluxe, nights, discountPercent, mealPackagePerNight);
        double baseCorp = deluxe.basePricePerNight * nights;
        double discountCorp = baseCorp * (discountPercent / 100.0);
        double mealsCorp = mealPackagePerNight * nights;
        System.out.println("\n--- Corporate Booking ---");
        System.out.println("Room: " + deluxe.type);
        System.out.println("Nights: " + nights);
        System.out.println("Base Price: Rs. " + baseCorp);
        System.out.println("Corporate Discount: " + discountPercent + "% (-Rs. " + discountCorp + ")");
        System.out.println("Meal Package: Rs. " + mealPackagePerNight + " x " + nights + " = Rs. " + mealsCorp);
        System.out.println("Total Price: Rs. " + corpTotal);

        // Wedding package
        int guestCount = 100;
        double decorationFee = 20000;
        double cateringPerGuest = 800;
        double wedTotal = BookingCalculator.calculateWedding(suite, nights, guestCount, decorationFee, cateringPerGuest);
        double baseWed = suite.basePricePerNight * nights;
        double cateringWed = cateringPerGuest * guestCount;
        System.out.println("\n--- Wedding Package ---");
        System.out.println("Room: " + suite.type);
        System.out.println("Nights: " + nights);
        System.out.println("Base Price: Rs. " + baseWed);
        System.out.println("Decoration Fee: Rs. " + decorationFee);
        System.out.println("Catering: Rs. " + cateringPerGuest + " x " + guestCount + " = Rs. " + cateringWed);
        System.out.println("Total Price: Rs. " + wedTotal);
    }
}
```

OUTPUT→

```
PS C:\Users\Ramesh\Personal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAVA-STEP\Weeks\Week 7\Assignment HW\Program1> cd "c:\Users\Ramesh\Per
al Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAVA-STEP\Weeks\Week 7\Assignment HW\Program1\" ; if ($?) { javac HotelReservationSystem.java }
if ($?) { java HotelReservationSystem }
--- Standard Booking ---
Room: Deluxe
Nights: 3
Base Price: Rs. 3000.0 x 3 = Rs. 9000.0

--- Seasonal Booking ---
Room: Suite
Nights: 3
Base Price: Rs. 5000.0 x 3 = Rs. 15000.0
Seasonal Multiplier: 1.2
Total Price: Rs. 18000.0

--- Corporate Booking ---
Room: Deluxe
Nights: 3
Base Price: Rs. 9000.0
Corporate Discount: 15.0% (-Rs. 1350.0)
Meal Package: Rs. 500.0 x 3 = Rs. 1500.0
Total Price: Rs. 9150.0

--- Wedding Package ---
Room: Suite
Nights: 3
Base Price: Rs. 15000.0
```

```
--- Wedding Package ---
Room: Suite
Nights: 3
Base Price: Rs. 15000.0
Decoration Fee: Rs. 20000.0
Catering: Rs. 800.0 x 100 = Rs. 80000.0
Total Price: Rs. 115000.0
```

PROBLEM 2: Online Learning Platform

Concept: Method Overriding

Create an educational content system where different course types display progress differently:

● Video courses show completion percentage and watch time

● Interactive courses show quiz scores and hands-on projects completed

● Reading courses show pages read and note-taking progress

● Certification courses show exam attempts and certification status

All courses share basic info (title, instructor, enrollment date) but track and display progress uniquely.

Hint: Common learning foundation, specialized progress tracking per course type!

PROGRAM→

```java
/*
 * Concept: Method Overriding
Create an educational content system where different course types display progress
differently:
• Video courses show completion percentage and watch time
• Interactive courses show quiz scores and hands-on projects completed
• Reading courses show pages read and note-taking progress
• Certification courses show exam attempts and certification status
All courses share basic info (title, instructor, enrollment date) but track and display
progress uniquely.
Hint: Common learning foundation, specialized progress tracking per course
type!
 */

import java.util.Date;

// Common learning foundation
class Course {
    protected String title;
    protected String instructor;
    protected Date enrollmentDate;

    public Course(String title, String instructor, Date enrollmentDate) {
        this.title = title;
        this.instructor = instructor;
        this.enrollmentDate = enrollmentDate;
    }

    public void displayProgress() {
        System.out.println("Progress for course: " + title);
    }
}
```

```java
// Video course
class VideoCourse extends Course {
    private double completionPercentage;
    private int watchTimeMinutes;

    public VideoCourse(String title, String instructor, Date enrollmentDate, double completionPercentage, int watchTimeMinutes) {
        super(title, instructor, enrollmentDate);
        this.completionPercentage = completionPercentage;
        this.watchTimeMinutes = watchTimeMinutes;
    }

    @Override
    public void displayProgress() {
        System.out.println("Video Course: " + title);
        System.out.println("Instructor: " + instructor);
        System.out.println("Enrolled on: " + enrollmentDate);
        System.out.println("Completion: " + completionPercentage + "%");
        System.out.println("Watch Time: " + watchTimeMinutes + " minutes");
    }
}

// Interactive course
class InteractiveCourse extends Course {
    private int quizScore;
    private int projectsCompleted;

    public InteractiveCourse(String title, String instructor, Date enrollmentDate, int quizScore, int projectsCompleted) {
        super(title, instructor, enrollmentDate);
        this.quizScore = quizScore;
        this.projectsCompleted = projectsCompleted;
    }

```

```
    @Override
    public void displayProgress() {
        System.out.println("Interactive Course: " + title);
        System.out.println("Instructor: " + instructor);
        System.out.println("Enrolled on: " + enrollmentDate);
        System.out.println("Quiz Score: " + quizScore);
        System.out.println("Projects Completed: " + projectsCompleted);
    }
}

// Reading course
class ReadingCourse extends Course {
    private int pagesRead;
    private int notesTaken;

    public ReadingCourse(String title, String instructor, Date enrollmentDate, int pagesRead, int notesTaken) {
        super(title, instructor, enrollmentDate);
        this.pagesRead = pagesRead;
        this.notesTaken = notesTaken;
    }

    @Override
    public void displayProgress() {
        System.out.println("Reading Course: " + title);
        System.out.println("Instructor: " + instructor);
        System.out.println("Enrolled on: " + enrollmentDate);
        System.out.println("Pages Read: " + pagesRead);
        System.out.println("Notes Taken: " + notesTaken);
    }
}
```


```
    @Override
    public void displayProgress() {
        System.out.println("Interactive Course: " + title);
        System.out.println("Instructor: " + instructor);
        System.out.println("Enrolled on: " + enrollmentDate);
        System.out.println("Quiz Score: " + quizScore);
        System.out.println("Projects Completed: " + projectsCompleted);
    }
}

// Reading course
class ReadingCourse extends Course {
    private int pagesRead;
    private int notesTaken;

    public ReadingCourse(String title, String instructor, Date enrollmentDate, int pagesRead, int notesTaken) {
        super(title, instructor, enrollmentDate);
        this.pagesRead = pagesRead;
        this.notesTaken = notesTaken;
    }

    @Override
    public void displayProgress() {
        System.out.println("Reading Course: " + title);
        System.out.println("Instructor: " + instructor);
        System.out.println("Enrolled on: " + enrollmentDate);
        System.out.println("Pages Read: " + pagesRead);
        System.out.println("Notes Taken: " + notesTaken);
    }
}
```


```
            vc.displayProgress();
            System.out.println();
            ic.displayProgress();
            System.out.println();
            rc.displayProgress();
            System.out.println();
            cc.displayProgress();
        }
    }
```

OUTPUT→


PS C:\Users\Ramesh\Personal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAVA-STEP\Weeks\Week 7\Assignment HW\Program2> cd "c:\Users\Ramesh\Personal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAVA-STEP\Weeks\Week 7\Assignment HW\Program2\" ; if ($?) { javac OnlineLearningPlatform.java } ; if ($?) { java OnlineLearningPlatform }
Video Course: Java Basics
Instructor: Dr. Smith
Enrolled on: Wed Sep 24 12:04:01 IST 2025
Completion: 75.5%
Watch Time: 120 minutes
Interactive Course: Python Interactive
Instructor: Ms. Lee
Enrolled on: Wed Sep 24 12:04:01 IST 2025
Quiz Score: 85
Projects Completed: 3
Reading Course: History 101
Instructor: Mr. Brown
Enrolled on: Wed Sep 24 12:04:01 IST 2025
Pages Read: 150
Notes Taken: 10
Certification Course: AWS Certification
Instructor: Mrs. Green
Enrolled on: Wed Sep 24 12:04:01 IST 2025
Exam Attempts: 2
Certification Status: Certified

PROBLEM 3: Transportation Fleet Management

Concept: Dynamic Method Dispatch

Design a city transport system with different vehicle types:

- **Buses follow fixed routes and track passenger capacity**
- **Taxis provide door-to-door service and calculate fare by distance**
- **Trains operate on schedules and manage multiple car capacity**
- **Bikes are available for short-distance eco-friendly trips**

Create a unified "dispatch" system where the same command produces appropriate transportation behavior based on vehicle type.

Hint: One dispatch call, many transport solutions - runtime polymorphism in action!

PROGRAM→

```java
/*
 * Concept: Dynamic Method Dispatch
Design a city transport system with different vehicle types:
• Buses follow fixed routes and track passenger capacity
• Taxis provide door-to-door service and calculate fare by distance
• Trains operate on schedules and manage multiple car capacity
• Bikes are available for short-distance eco-friendly trips
Create a unified "dispatch" system where the same command produces appropriate
transportation behavior based on vehicle type.
Hint: One dispatch call, many transport solutions - runtime polymorphism in
action!
 */

abstract class Vehicle {
    abstract void dispatch();
}

class Bus extends Vehicle {
    private String route;
    private int passengerCapacity;

    Bus(String route, int passengerCapacity) {
        this.route = route;
        this.passengerCapacity = passengerCapacity;
    }

    void dispatch() {
        System.out.println("Bus dispatched on route " + route +
            " with capacity " + passengerCapacity + " passengers.");
    }
}

class Taxi extends Vehicle {
    private String destination;
    private double distance;

    Taxi(String destination, double distance) {
        this.destination = destination;
        this.distance = distance;
    }

    void dispatch() {
        double fare = distance * 15; // Example fare calculation
        System.out.println("Taxi dispatched to " + destination +
            ". Distance: " + distance + " km. Fare: Rs." + fare);
    }
}

class Train extends Vehicle {
    private String schedule;
    private int carCapacity;

    Train(String schedule, int carCapacity) {
        this.schedule = schedule;
        this.carCapacity = carCapacity;
    }

    void dispatch() {
        System.out.println("Train dispatched as per schedule " + schedule +
            " with " + carCapacity + " cars.");
    }
}
```

```java
class Bike extends Vehicle {
    private String pickupPoint;
    private String dropPoint;

    Bike(String pickupPoint, String dropPoint) {
        this.pickupPoint = pickupPoint;
        this.dropPoint = dropPoint;
    }

    void dispatch() {
        System.out.println("Bike dispatched from " + pickupPoint +
            " to " + dropPoint + ". Eco-friendly short trip!");
    }
}

public class TransporationFleetManagement {
    Run main | Debug main
    public static void main(String[] args) {
        Vehicle v;

        v = new Bus("Route 21A", 50);
        v.dispatch();

        v = new Taxi("Central Mall", 12.5);
        v.dispatch();

        v = new Train("10:30 AM Express", 8);
        v.dispatch();

        v = new Bike("Park Street", "Library");
        v.dispatch();
    }
}
```

OUTPUT→

```
PS C:\Users\Ramesh\Personal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAVA-STEP\Weeks\Week 7\Assignment HW\Progra
m3> cd "c:\Users\Ramesh\Personal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAVA-STEP\Weeks\Week 7\Assignment HW\P
rogram3\" ; if ($?) { javac TransporationFleetManagement.java } ; if ($?) { java TransporationFleetManagement }
Bus dispatched on route Route 21A with capacity 50 passengers.
Taxi dispatched to Central Mall. Distance: 12.5 km. Fare: Rs.187.5
Train dispatched as per schedule 10:30 AM Express with 8 cars.
Bike dispatched from Park Street to Library. Eco-friendly short trip!
```

PROBLEM 4: Hospital Management System

Concept: Upcasting

Build a hospital system managing different types of medical staff:

- **Doctors can diagnose patients, prescribe medicine, and perform surgeries**
- **Nurses can administer medicine, monitor patients, and assist procedures**
- **Technicians can operate equipment, run tests, and maintain instruments**
- **Administrators can schedule appointments and manage records**

Design a general "MedicalStaff" system for common operations like shift scheduling, ID card access, and payroll processing.

Hint: Different specialties, common professional needs - think institutional level!

PROGRAM→

```java
/*
 * Concept: Upcasting
Build a hospital system managing different types of medical staff:
• Doctors can diagnose patients, prescribe medicine, and perform surgeries
• Nurses can administer medicine, monitor patients, and assist procedures
• Technicians can operate equipment, run tests, and maintain instruments
• Administrators can schedule appointments and manage records
Design a general "MedicalStaff" system for common operations like shift scheduling, ID
card access, and payroll processing.
Hint: Different specialties, common professional needs - think institutional level!
 */

abstract class MedicalStaff {
    protected String name;
    protected int staffId;

    MedicalStaff(String name, int staffId) {
        this.name = name;
        this.staffId = staffId;
    }

    void shiftScheduling() {
        System.out.println(name + " (ID: " + staffId + ") shift scheduled.");
    }

    void idCardAccess() {
        System.out.println(name + " (ID: " + staffId + ") has ID card access.");
    }

    void payrollProcessing() {
        System.out.println(name + " (ID: " + staffId + ") payroll processed.");
    }

    abstract void performDuties();
}

class Doctor extends MedicalStaff {
    Doctor(String name, int staffId) {
        super(name, staffId);
    }

    void diagnosePatient() {
        System.out.println(name + " is diagnosing a patient.");
    }

    void prescribeMedicine() {
        System.out.println(name + " is prescribing medicine.");
    }

    void performSurgery() {
        System.out.println(name + " is performing surgery.");
    }

    @Override
    void performDuties() {
        diagnosePatient();
        prescribeMedicine();
        performSurgery();
    }
}

class Nurse extends MedicalStaff {
    Nurse(String name, int staffId) {
        super(name, staffId);
    }
```

```
    void administerMedicine() {
        System.out.println(name + " is administering medicine.");
    }

    void monitorPatient() {
        System.out.println(name + " is monitoring a patient.");
    }

    void assistProcedure() {
        System.out.println(name + " is assisting in a procedure.");
    }

    @Override
    void performDuties() {
        administerMedicine();
        monitorPatient();
        assistProcedure();
    }
}

class Technician extends MedicalStaff {
    Technician(String name, int staffId) {
        super(name, staffId);
    }

    void operateEquipment() {
        System.out.println(name + " is operating equipment.");
    }

    void runTests() {
        System.out.println(name + " is running tests.");
    }
    void maintainInstruments() {
        System.out.println(name + " is maintaining instruments.");
    }

    @Override
    void performDuties() {
        operateEquipment();
        runTests();
        maintainInstruments();
    }
}

class Administrator extends MedicalStaff {
    Administrator(String name, int staffId) {
        super(name, staffId);
    }

    void scheduleAppointments() {
        System.out.println(name + " is scheduling appointments.");
    }

    void manageRecords() {
        System.out.println(name + " is managing records.");
    }

    @Override
    void performDuties() {
        scheduleAppointments();
        manageRecords();
    }
}
```

```java
public class HospitalManagementSystem {
    Run main | Debug main
    public static void main(String[] args) {
        MedicalStaff[] staffList = new MedicalStaff[4];
        staffList[0] = new Doctor("Dr. Smith", 101);
        staffList[1] = new Nurse("Nurse Alice", 201);
        staffList[2] = new Technician("Tech Bob", 301);
        staffList[3] = new Administrator("Admin Carol", 401);

        for (MedicalStaff staff : staffList) {
            System.out.println("\n--- " + staff.getClass().getSimpleName() + " ---");
            staff.shiftScheduling();
            staff.idCardAccess();
            staff.payrollProcessing();
            staff.performDuties();
        }
    }
}
```

OUTPUT→

```
PS C:\Users\Ramesh\Personal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAVA-STEP\Weeks\Week 7\Assignment HW\Prog
m4> cd "c:\Users\Ramesh\Personal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAVA-STEP\Weeks\Week 7\Assignment HW
rogram4\" ; if ($?) { javac HospitalManagementSystem.java } ; if ($?) { java HospitalManagementSystem }

--- Doctor ---
Dr. Smith (ID: 101) shift scheduled.
Dr. Smith (ID: 101) has ID card access.
Dr. Smith (ID: 101) payroll processed.
Dr. Smith is diagnosing a patient.
Dr. Smith is prescribing medicine.
Dr. Smith is performing surgery.

--- Nurse ---
Nurse Alice (ID: 201) shift scheduled.
Nurse Alice (ID: 201) has ID card access.
Nurse Alice (ID: 201) payroll processed.
Nurse Alice is administering medicine.
Nurse Alice is monitoring a patient.
Nurse Alice is assisting in a procedure.

--- Technician ---
Tech Bob (ID: 301) shift scheduled.
Tech Bob (ID: 301) has ID card access.
Tech Bob (ID: 301) payroll processed.
Tech Bob is operating equipment.
Tech Bob is running tests.
Tech Bob is maintaining instruments.
```

```
--- Administrator ---
Admin Carol (ID: 401) shift scheduled.
Admin Carol (ID: 401) has ID card access.
Admin Carol (ID: 401) payroll processed.
Admin Carol is scheduling appointments.
Admin Carol is managing records.
```

PROBLEM 5: Digital Art Gallery

Concept: Downcasting

Create an art gallery system handling different artwork types:

- **Paintings have brush techniques, color palettes, and frame specifications**
- **Sculptures have material composition, dimensions, and lighting requirements**
- **Digital art has resolution, file formats, and interactive elements**
- **Photography has camera settings, editing details, and print specifications**

Sometimes curators need access to specific artwork features for exhibition planning.

Hint: From general art piece to specific medium - unlock the details when needed!

PROGRAM→

```java
/*
 * Concept: Downcasting
Create an art gallery system handling different artwork types:
• Paintings have brush techniques, color palettes, and frame specifications
• Sculptures have material composition, dimensions, and lighting requirements
• Digital art has resolution, file formats, and interactive elements
• Photography has camera settings, editing details, and print specifications
Sometimes curators need access to specific artwork features for exhibition planning.
Hint: From general art piece to specific medium - unlock the details when needed!
 */

abstract class ArtPiece {
    String title;
    String artist;

    ArtPiece(String title, String artist) {
        this.title = title;
        this.artist = artist;
    }

    void displayBasicInfo() {
        System.out.println("Title: " + title + ", Artist: " + artist);
    }
}

class Painting extends ArtPiece {
    String brushTechnique;
    String colorPalette;
    String frameSpec;

    Painting(String title, String artist, String brushTechnique, String colorPalette, String frameSpec) {
        super(title, artist);
        this.brushTechnique = brushTechnique;
```

```
        this.colorPalette = colorPalette;
        this.frameSpec = frameSpec;
    }

    void displayPaintingDetails() {
        System.out.println("Brush Technique: " + brushTechnique);
        System.out.println("Color Palette: " + colorPalette);
        System.out.println("Frame Specification: " + frameSpec);
    }
}

class Sculpture extends ArtPiece {
    String material;
    String dimensions;
    String lightingReq;

    Sculpture(String title, String artist, String material, String dimensions, String lightingReq) {
        super(title, artist);
        this.material = material;
        this.dimensions = dimensions;
        this.lightingReq = lightingReq;
    }

    void displaySculptureDetails() {
        System.out.println("Material Composition: " + material);
        System.out.println("Dimensions: " + dimensions);
        System.out.println("Lighting Requirements: " + lightingReq);
    }
}
```

```
class DigitalArt extends ArtPiece {
    String resolution;
    String fileFormat;
    String interactiveElements;

    DigitalArt(String title, String artist, String resolution, String fileFormat, String interactiveElements) {
        super(title, artist);
        this.resolution = resolution;
        this.fileFormat = fileFormat;
        this.interactiveElements = interactiveElements;
    }

    void displayDigitalArtDetails() {
        System.out.println("Resolution: " + resolution);
        System.out.println("File Format: " + fileFormat);
        System.out.println("Interactive Elements: " + interactiveElements);
    }
}

class Photography extends ArtPiece {
    String cameraSettings;
    String editingDetails;
    String printSpec;

    Photography(String title, String artist, String cameraSettings, String editingDetails, String printSpec) {
        super(title, artist);
        this.cameraSettings = cameraSettings;
        this.editingDetails = editingDetails;
        this.printSpec = printSpec;
    }

    void displayPhotographyDetails() {
```

```
    void displayPhotographyDetails() {
        System.out.println("Camera Settings: " + cameraSettings);
        System.out.println("Editing Details: " + editingDetails);
        System.out.println("Print Specification: " + printSpec);
    }
}

public class DigitalArtGallery {
    Run main | Debug main
    public static void main(String[] args) {
        // Create an array of general ArtPiece references
        ArtPiece[] gallery = new ArtPiece[4];
        gallery[0] = new Painting("Sunset Bliss", "A. Kumar", "Impressionist", "Warm Tones", "Ornate Gold Frame");
        gallery[1] = new Sculpture("The Thinker", "B. Singh", "Bronze", "2m x 1m x 1m", "Spotlight");
        gallery[2] = new DigitalArt("Virtual Dreams", "C. Rao", "4K", "PNG", "Touch Responsive");
        gallery[3] = new Photography("City Lights", "D. Patel", "ISO 800, f/2.8", "HDR, Cropped", "Glossy A3");

        for (ArtPiece art : gallery) {
            art.displayBasicInfo();

            // Downcasting to access specific details
            if (art instanceof Painting) {
                System.out.println("-- Painting Details --");
                ((Painting) art).displayPaintingDetails();
            } else if (art instanceof Sculpture) {
                System.out.println("-- Sculpture Details --");
                ((Sculpture) art).displaySculptureDetails();
            } else if (art instanceof DigitalArt) {
                System.out.println("-- Digital Art Details --");
                ((DigitalArt) art).displayDigitalArtDetails();
            } else if (art instanceof Photography) {
                System.out.println("-- Photography Details --");
```

OUTPUT→

```
PS C:\Users\Ramesh\Personal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAVA-STEP\Weeks\Week 7\Assignment HW\Progr
m5> cd "c:\Users\Ramesh\Personal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAVA-STEP\Weeks\Week 7\Assignment HW\
rogram5\" ; if ($?) { javac DigitalArtGallery.java } ; if ($?) { java DigitalArtGallery }
Title: Sunset Bliss, Artist: A. Kumar
-- Painting Details --
Brush Technique: Impressionist
Color Palette: Warm Tones
Frame Specification: Ornate Gold Frame

Title: The Thinker, Artist: B. Singh
-- Sculpture Details --
Material Composition: Bronze
Dimensions: 2m x 1m x 1m
Lighting Requirements: Spotlight

Title: Virtual Dreams, Artist: C. Rao
-- Digital Art Details --
Resolution: 4K
File Format: PNG
Interactive Elements: Touch Responsive

Title: City Lights, Artist: D. Patel
-- Photography Details --
Camera Settings: ISO 800, f/2.8
Editing Details: HDR, Cropped
Print Specification: Glossy A3
```

PROBLEM 6: Smart Home Automation

Concept: Safe Downcasting with instanceof

Design a home automation system controlling various smart devices:

● Smart TVs manage channels, volume, and streaming apps

● Smart thermostats control temperature, humidity, and energy saving modes

● Smart security systems handle cameras, alarms, and access controls

● Smart kitchen appliances manage cooking times, temperatures, and recipes

Process mixed device collections safely, applying appropriate controls without system crashes.

Hint: Identify before you control - each device has its own smart features!

PROGRAM→

```java
/*
 * Concept: Safe Downcasting with instanceof
Design a home automation system controlling various smart devices:
● Smart TVs manage channels, volume, and streaming apps
● Smart thermostats control temperature, humidity, and energy saving modes
● Smart security systems handle cameras, alarms, and access controls
● Smart kitchen appliances manage cooking times, temperatures, and recipes
Process mixed device collections safely, applying appropriate controls without system
crashes.
Hint: Identify before you control - each device has its own smart features!
 */

class SmartTV {
    void changeChannel(int channel) {
        System.out.println("SmartTV: Changing to channel " + channel);
    }
    void adjustVolume(int volume) {
        System.out.println("SmartTV: Setting volume to " + volume);
    }
    void openStreamingApp(String app) {
        System.out.println("SmartTV: Opening streaming app " + app);
    }
}

class SmartThermostat {
    void setTemperature(double temp) {
        System.out.println("Thermostat: Setting temperature to " + temp + "°C");
    }
    void setHumidity(int humidity) {
        System.out.println("Thermostat: Setting humidity to " + humidity + "%");
    }
    void enableEnergySavingMode() {
        System.out.println("Thermostat: Energy saving mode enabled");
    }
```

```java
class SmartSecuritySystem {
    void activateCamera() {
        System.out.println("Security: Camera activated");
    }
    void soundAlarm() {
        System.out.println("Security: Alarm sounding!");
    }
    void controlAccess(String user) {
        System.out.println("Security: Access control for " + user);
    }
}

class SmartKitchenAppliance {
    void setCookingTime(int minutes) {
        System.out.println("Kitchen: Cooking time set to " + minutes + " minutes");
    }
    void setCookingTemperature(int temp) {
        System.out.println("Kitchen: Cooking temperature set to " + temp + "°C");
    }
    void selectRecipe(String recipe) {
        System.out.println("Kitchen: Recipe selected - " + recipe);
    }
}

public class SmartHomeAutomation {
    Run main | Debug main
    public static void main(String[] args) {
        // Mixed collection of smart devices
        Object[] devices = {
            new SmartTV(),
            new SmartThermostat(),
            new SmartSecuritySystem(),
            new SmartKitchenAppliance(),
            new SmartKitchenAppliance(),
            new SmartTV(),
            new SmartKitchenAppliance()
        };

        for (Object device : devices) {
            // Safe downcasting using instanceof
            if (device instanceof SmartTV) {
                SmartTV tv = (SmartTV) device;
                tv.changeChannel(5);
                tv.adjustVolume(15);
                tv.openStreamingApp("Netflix");
            } else if (device instanceof SmartThermostat) {
                SmartThermostat thermostat = (SmartThermostat) device;
                thermostat.setTemperature(22.5);
                thermostat.setHumidity(45);
                thermostat.enableEnergySavingMode();
            } else if (device instanceof SmartSecuritySystem) {
                SmartSecuritySystem security = (SmartSecuritySystem) device;
                security.activateCamera();
                security.soundAlarm();
                security.controlAccess("Alice");
            } else if (device instanceof SmartKitchenAppliance) {
                SmartKitchenAppliance kitchen = (SmartKitchenAppliance) device;
                kitchen.setCookingTime(30);
                kitchen.setCookingTemperature(180);
                kitchen.selectRecipe("Pasta");
            } else {
                System.out.println("Unknown device detected.");
            }
            System.out.println("---");
```

OUTPUT→

```
PS C:\Users\Ramesh\Personal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAVA-STEP\Weeks\Week 7\Assignment HW\Pro
m6> cd "c:\Users\Ramesh\Personal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAVA-STEP\Weeks\Week 7\Assignment H
rogram6\" ; if ($?) { javac SmartHomeAutomation.java } ; if ($?) { java SmartHomeAutomation }
SmartTV: Changing to channel 5
SmartTV: Setting volume to 15
SmartTV: Opening streaming app Netflix
---
Thermostat: Setting temperature to 22.5°C
Thermostat: Setting humidity to 45%
Thermostat: Energy saving mode enabled
---
Security: Camera activated
Security: Alarm sounding!
Security: Access control for Alice
---
Kitchen: Cooking time set to 30 minutes
Kitchen: Cooking temperature set to 180°C
Kitchen: Recipe selected - Pasta
---
SmartTV: Changing to channel 5
SmartTV: Setting volume to 15
SmartTV: Opening streaming app Netflix
---
Kitchen: Cooking time set to 30 minutes
Kitchen: Cooking temperature set to 180°C
Kitchen: Recipe selected - Pasta
---
```