Week 7 - S7 - Core OOP - Polymorphism - Lab Problem

Name:Ramesh Harisabapathi Chettiar

Date of Submission:24/09/25

PROBLEM 1: Food Delivery App

Concept: Method Overloading

You're creating a food ordering system. Design a class that can calculate delivery charges in different ways:

- **Basic delivery (just distance)**
- **Premium delivery (distance + priority fee)**
- **Group delivery (distance + number of orders discount)**
- **Festival special (distance + discount percentage + free delivery over certain amount)**

Each calculation should show a different message about the delivery cost breakdown.

Hint: Same method name, different parameters - let Java pick the right one!

PROGRAM→

```java
/*
 * You're creating a food ordering system. Design a class that can calculate delivery
charges in different ways:
• Basic delivery (just distance)
• Premium delivery (distance + priority fee)
• Group delivery (distance + number of orders discount)
• Festival special (distance + discount percentage + free delivery over certain
amount)
Each calculation should show a different message about the delivery cost breakdown.
 */

class BasicDelivery {
    public void calculate(double distance) {
        double cost = distance * 10; // Rs.10 per km
        System.out.println("Basic Delivery: Distance = " + distance + " km, Cost = Rs." + cost);
    }
}

class PremiumDelivery {
    public void calculate(double distance, double priorityFee) {
        double cost = distance * 10 + priorityFee;
        System.out.println("Premium Delivery: Distance = " + distance + " km, Priority Fee = Rs." + priorityFee + ", Total Cost = Rs." + co
    }
}

class GroupDelivery {
    public void calculate(double distance, int numOrders) {
        double base = distance * 10;
        double discount = numOrders * 5; // Rs.5 discount per order
        double cost = base - discount;
        if (cost < 0) cost = 0;
        System.out.println("Group Delivery: Distance = " + distance + " km, Orders = " + numOrders + ", Discount = Rs." + discount + ", Tot
    }
```

```java
class FestivalSpecial {
    public void calculate(double distance, double discountPercent, double orderAmount) {
        double base = distance * 10;
        double discount = base * (discountPercent / 100.0);
        double cost = base - discount;
        if (orderAmount >= 500) {
            System.out.println("Festival Special: Order Amount = Rs." + orderAmount + " (Free Delivery!)");
            cost = 0;
        } else {
            System.out.println("Festival Special: Distance = " + distance + " km, Discount = " + discountPercent + "%, Discount Amount = Rs
        }
    }
}

public class FoodDelivery {
    Run main | Debug main
    public static void main(String[] args) {
        BasicDelivery bd = new BasicDelivery();
        PremiumDelivery pd = new PremiumDelivery();
        GroupDelivery gd = new GroupDelivery();
        FestivalSpecial fs = new FestivalSpecial();

        // Basic Delivery
        bd.calculate(5);

        // Premium Delivery
        pd.calculate(5, 50);

        // Group Delivery
        gd.calculate(5, 3);

        // Festival Special
        fs.calculate(5, 20, 400);
```

OUTPUT→

```
PS C:\Users\Ramesh\Personal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAVA-STEP\Weeks\Week 7\Lab Problems\Program1> cd "c:\Users\Ramesh
ersonal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAVA-STEP\Weeks\Week 7\Lab Problems\Program1\" ; if ($?) { javac FoodDelivery.java }
if ($?) { java FoodDelivery }
Basic Delivery: Distance = 5.0 km, Cost = Rs.50.0
Premium Delivery: Distance = 5.0 km, Priority Fee = Rs.50.0, Total Cost = Rs.100.0
Group Delivery: Distance = 5.0 km, Orders = 3, Discount = Rs.15.0, Total Cost = Rs.35.0
Festival Special: Distance = 5.0 km, Discount = 20.0%, Discount Amount = Rs.10.0, Total Cost = Rs.40.0
Festival Special: Order Amount = Rs.600.0 (Free Delivery!)
```

PROBLEM 2: Social Media Feed

Concept: Method Overriding

Build a social media post system where different platforms display posts differently:

- **Instagram posts show with hashtags and likes**
- **Twitter posts show with character count and retweets**
- **LinkedIn posts show with professional formatting and connections**

All posts share common info (author, content, time) but display uniquely for each platform.

Hint: Parent class defines the structure, child classes customize the display!

PROGRAM→

SocialMediaPost.java

```java
/*
 * Build a social media post system where different platforms display posts differently:
• Instagram posts show with hashtags and likes
• Twitter posts show with character count and retweets
• LinkedIn posts show with professional formatting and connections
All posts share common info (author, content, time) but display uniquely for each
platform.
Hint: Parent class defines the structure, child classes customize the display!
 */

class CommonInfo {
    protected String author;
    protected String content;
    protected int time;

    CommonInfo(String author, String content, int time) {
        this.author = author;
        this.content = content;
        this.time = time;
    }

    public void display() {
        System.out.println("=============COMMON INFO=============");
        System.out.println("Author --> " + author);
        System.out.println("Content --> " + content);
        System.out.println("Time --> " + time);
    }
}

class InstagramPost extends CommonInfo {
    private int hashtags;
    private int likes;

    InstagramPost(String author, String content, int time, int hashtags, int likes) {
        super(author, content, time);
        this.hashtags = hashtags;
        this.likes = likes;
    }

    public void showPosts() {
        display();
        System.out.println("Instagram Post --> " + hashtags + " hashtags and " + likes + " likes!");
    }
}

class TwitterPost extends CommonInfo {
    private int characterCount;
    private int retweets;

    TwitterPost(String author, String content, int time, int characterCount, int retweets) {
        super(author, content, time);
        this.characterCount = characterCount;
        this.retweets = retweets;
    }

    public void showPosts() {
        display();
        System.out.println("Twitter Post --> " + characterCount + " characters and " + retweets + " retweets!");
    }
}

class LinkedInPost extends CommonInfo {
    private int connections;

    LinkedInPost(String author, String content, int time, int connections) {
```

```java
    LinkedInPost(String author, String content, int time, int connections) {
        super(author, content, time);
        this.connections = connections;
    }

    public void showPosts() {
        display();
        System.out.println("LinkedIn Post --> Professional formatting and " + connections + " connections!");
    }
}

public class SocialMediaPost {
    Run main | Debug main
    public static void main(String[] args) {
        InstagramPost ig = new InstagramPost("Ananya", "Enjoying the sunset!", 18, 5, 120);
        TwitterPost tw = new TwitterPost("Vijay", "Java is awesome!", 19, 50, 30);
        LinkedInPost ln = new LinkedInPost("Kareena", "Excited to join SRM!", 20, 500);

        ig.showPosts();
        System.out.println();
        tw.showPosts();
        System.out.println();
        ln.showPosts();
    }
}
```

OUTPUT→

```
PS C:\Users\Ramesh\Personal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAVA-STEP\Weeks\Week 7\Lab Problems\Program2> cd "c:\Users\Ramesh\P
ersonal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAVA-STEP\Weeks\Week 7\Lab Problems\Program2\" ; if ($?) { javac SocialMediaPost.java }
 ; if ($?) { java SocialMediaPost }
============COMMON INFO============
Author --> Ananya
Content --> Enjoying the sunset!
Time --> 18
Instagram Post --> 5 hashtags and 120 likes!

============COMMON INFO============
Author --> Vijay
Content --> Java is awesome!
Time --> 19
Twitter Post --> 50 characters and 30 retweets!

============COMMON INFO============
Author --> Kareena
Content --> Excited to join SRM!
Time --> 20
LinkedIn Post --> Professional formatting and 500 connections!
```

PROBLEM 3: Gaming Character System

Concept: Dynamic Method Dispatch

Create a battle system with different character types:

- **Warriors attack with weapons and have high defense**
- **Mages cast spells and use mana**
- **Archers shoot arrows with long-range damage**

Design it so the same "attack" command produces different results based on the character type, even when stored in a mixed army array.

Hint: Same reference, different objects - let runtime decide the behavior!

PROGRAM→

```java
/*
 * Create a battle system with different character types:
• Warriors attack with weapons and have high defense
• Mages cast spells and use mana
• Archers shoot arrows with long-range damage
Design it so the same "attack" command produces different results based on the
character type, even when stored in a mixed army array.
Hint: Same reference, different objects - let runtime decide the behavior!
*/

abstract class Character {
    String name;
    Character(String name) {
        this.name = name;
    }
    abstract void attack();
}

class Warrior extends Character {
    Warrior(String name) {
        super(name);
    }
    void attack() {
        System.out.println(name + " swings a sword with high defense!");
    }
}

class Mage extends Character {
    Mage(String name) {
        super(name);
    }
    void attack() {
        System.out.println(name + " casts a powerful spell using mana!");
    }
}

class Archer extends Character {
    Archer(String name) {
        super(name);
    }
    void attack() {
        System.out.println(name + " shoots a long-range arrow!");
    }
}

public class GamingCharacterSystem {
    Run main | Debug main
    public static void main(String[] args) {
        Character[] army = new Character[3];
        army[0] = new Warrior("Thor");
        army[1] = new Mage("Merlin");
        army[2] = new Archer("Robin");

        for (Character c : army) {
            c.attack();
        }
    }
}
```

OUTPUT→

```
PS C:\Users\Ramesh\Personal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAVA-STEP\Weeks\Week 7\Lab Problems\Program3> cd "c:\Users\Ramesh\P
ersonal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAVA-STEP\Weeks\Week 7\Lab Problems\Program3\" ; if ($?) { javac GamingCharacterSystem.
java } ; if ($?) { java GamingCharacterSystem }
Thor swings a sword with high defense!
Merlin casts a powerful spell using mana!
Robin shoots a long-range arrow!
```

PROBLEM 4: University Library System

Concept: Upcasting

Design a library system with different types of users:

- **Students can borrow books and access computers**
- **Faculty can reserve books and access research databases**
- **Guests can only browse books**

Create a general "LibraryUser" system that can handle any user type for common operations like entry logging and basic info display.

Hint: Think bigger picture - store specialists as generalists safely!

PROGRAM→

```java
/*
 * Design a library system with different types of users:
• Students can borrow books and access computers
• Faculty can reserve books and access research databases
• Guests can only browse books
Create a general "LibraryUser" system that can handle any user type for common
operations like entry logging and basic info display.
Hint: Think bigger picture - store specialists as generalists safely!
 */

import java.util.*;

class LibraryUser {
    protected String name;
    protected String userType;

    public LibraryUser(String name, String userType) {
        this.name = name;
        this.userType = userType;
    }

    public void logEntry() {
        System.out.println(name + " (" + userType + ") entered the library.");
    }

    public void displayInfo() {
        System.out.println("Name: " + name + ", Type: " + userType);
    }

    // Common operation: browse books
    public void browseBooks() {
        System.out.println(name + " is browsing books.");
    }
}
```

```java
class Student extends LibraryUser {
    public Student(String name) {
        super(name, "Student");
    }

    public void borrowBook(String book) {
        System.out.println(name + " borrowed the book: " + book);
    }

    public void accessComputer() {
        System.out.println(name + " is accessing a computer.");
    }
}

class Faculty extends LibraryUser {
    public Faculty(String name) {
        super(name, "Faculty");
    }

    public void reserveBook(String book) {
        System.out.println(name + " reserved the book: " + book);
    }

    public void accessResearchDatabase() {
        System.out.println(name + " is accessing research databases.");
    }
}

class Guest extends LibraryUser {
    public Guest(String name) {
        super(name, "Guest");
    }
}
```

```java
public class UniversityLibrarySystem {

    public void addUser(LibraryUser user) {
        users.add(user);
        user.logEntry();
    }

    public void displayAllUsers() {
        System.out.println("\nLibrary Users:");
        for (LibraryUser user : users) {
            user.displayInfo();
        }
    }

    Run main | Debug main
    public static void main(String[] args) {
        UniversityLibrarySystem system = new UniversityLibrarySystem();

        // Create users
        Student s1 = new Student("Amar");
        Faculty f1 = new Faculty("Akbar");
        Guest g1 = new Guest("Anthony");

        // Add users to system
        system.addUser(s1);
        system.addUser(f1);
        system.addUser(g1);

        // Common operations
        s1.browseBooks();
        f1.browseBooks();
        g1.browseBooks();

        // Specialist operations
        s1.borrowBook("Java Programming");
        s1.accessComputer();

        f1.reserveBook("Data Structures");
        f1.accessResearchDatabase();

        // Display all users
        system.displayAllUsers();
    }
}
```

OUTPUT→

```
PS C:\Users\Ramesh\Personal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAVA-STEP\Weeks\Week 7\Lab Problems\Program4> cd
ersonal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAVA-STEP\Weeks\Week 7\Lab Problems\Program4\" ; if ($?) { javac Univ
m.java } ; if ($?) { java UniversityLibrarySystem }
Amar (Student) entered the library.
Akbar (Faculty) entered the library.
Anthony (Guest) entered the library.
Amar is browsing books.
Akbar is browsing books.
Anthony is browsing books.
Amar borrowed the book: Java Programming
Amar is accessing a computer.
Akbar reserved the book: Data Structures
Akbar is accessing research databases.

Library Users:
Name: Amar, Type: Student
Name: Akbar, Type: Faculty
Name: Anthony, Type: Guest
PS C:\Users\Ramesh\Personal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAVA-STEP\Weeks\Week 7\Lab Problems\Program4>
```

PROBLEM 5: Movie Streaming Platform

Concept: Downcasting

Build a streaming service that handles different content types:

- **Movies have ratings, duration, and subtitle options**
- **TV Series have seasons, episodes, and next episode suggestions**
- **Documentaries have educational tags and related content**

Sometimes you need to access specific features based on what the user is actually watching.

Hint: Go from general to specific - but be careful, not everything is what it seems!

PROGRAM→

```java
/*
 * Build a streaming service that handles different content types:
• Movies have ratings, duration, and subtitle options
• TV Series have seasons, episodes, and next episode suggestions
• Documentaries have educational tags and related content
Sometimes you need to access specific features based on what the user is actually
watching.
Hint: Go from general to specific - but be careful, not everything is what it seems!
*/

abstract class Content {
    String title;
    Content(String title) {
        this.title = title;
    }
    void displayInfo() {
        System.out.println("Title: " + title);
    }
}

class Movie extends Content {
    double rating;
    int duration; // in minutes
    boolean hasSubtitles;

    Movie(String title, double rating, int duration, boolean hasSubtitles) {
        super(title);
        this.rating = rating;
        this.duration = duration;
        this.hasSubtitles = hasSubtitles;
    }

```

```java
class TVSeries extends Content {
    int seasons;
    int episodes;
    String nextEpisodeSuggestion;

    TVSeries(String title, int seasons, int episodes, String nextEpisodeSuggestion) {
        super(title);
        this.seasons = seasons;
        this.episodes = episodes;
        this.nextEpisodeSuggestion = nextEpisodeSuggestion;
    }

    void showSeriesFeatures() {
        System.out.println("Seasons: " + seasons);
        System.out.println("Episodes: " + episodes);
        System.out.println("Next Episode: " + nextEpisodeSuggestion);
    }
}

class Documentary extends Content {
    String[] educationalTags;
    String relatedContent;

    Documentary(String title, String[] educationalTags, String relatedContent) {
        super(title);
        this.educationalTags = educationalTags;
```

```java

    void showDocumentaryFeatures() {
        System.out.print("Educational Tags: ");
        for (String tag : educationalTags) {
            System.out.print(tag + " ");
        }
        System.out.println();
        System.out.println("Related Content: " + relatedContent);
    }
}

public class MovieStreamingPlatform {
    Run main | Debug main
    public static void main(String[] args) {
        Content[] watchList = new Content[3];
        watchList[0] = new Movie("Inception", 8.8, 148, true);
        watchList[1] = new TVSeries("Stranger Things", 4, 34, "Season 4, Episode 2");
        watchList[2] = new Documentary("Planet Earth", new String[]{"Nature", "Wildlife"}, "Blue Planet");

        for (Content content : watchList) {
            content.displayInfo();
            // Access specific features based on actual type
            if (content instanceof Movie) {
                ((Movie) content).showMovieFeatures();
            } else if (content instanceof TVSeries) {
                ((TVSeries) content).showSeriesFeatures();
            } else if (content instanceof Documentary) {
                ((Documentary) content).showDocumentaryFeatures();
            }
            System.out.println("--------------------------");
        }
    }
```

Sorry, something went wrong activating IntelliCo

OUTPUT→

```
PS C:\Users\Ramesh\Personal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAVA-STEP\Weeks\Week 7\Lab Problems\Program5> cd "c:\Users\Ramesh\P
ersonal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAVA-STEP\Weeks\Week 7\Lab Problems\Program5\" ; if ($?) { javac MovieStreamingPlatform
.java } ; if ($?) { java MovieStreamingPlatform }
Title: Inception
Movie Rating: 8.8
Duration: 148 mins
Subtitles Available: Yes
--------------------------
Title: Stranger Things
Seasons: 4
Episodes: 34
Next Episode: Season 4, Episode 2
--------------------------
Title: Planet Earth
Educational Tags: Nature Wildlife
Related Content: Blue Planet
--------------------------
```

PROBLEM 6: Smart Campus IoT System

Concept: Safe Downcasting with instanceof

Create a campus management system with different smart devices:

- **Smart classrooms control lighting, AC, and projectors**
- **Smart labs manage equipment and safety systems**
- **Smart libraries track occupancy and book availability**

Process mixed device collections safely, applying the right controls to each device type without crashing.

Hint: Check first, cast second - safety matters in the real world!

PROGRAM→

```java
/*
 * Create a campus management system with different smart devices:
• Smart classrooms control lighting, AC, and projectors
• Smart labs manage equipment and safety systems
• Smart libraries track occupancy and book availability
Process mixed device collections safely, applying the right controls to each device type
without crashing.
Hint: Check first, cast second - safety matters in the real world!
 */

interface SmartDevice {
    void status();
}

class SmartClassroom implements SmartDevice {
    private boolean lightsOn = false;
    private boolean acOn = false;
    private boolean projectorOn = false;

    public void controlLights(boolean on) {
        lightsOn = on;
        System.out.println("Classroom lights " + (on ? "ON" : "OFF"));
    }

    public void controlAC(boolean on) {
        acOn = on;
        System.out.println("Classroom AC " + (on ? "ON" : "OFF"));
    }

    public void controlProjector(boolean on) {
        projectorOn = on;
        System.out.println("Classroom projector " + (on ? "ON" : "OFF"));
    }

    public void status() {
        System.out.println("Classroom Status: Lights=" + lightsOn + ", AC=" + acOn + ", Projector=" + projectorOn);
    }
}

class SmartLab implements SmartDevice {
    private boolean equipmentOn = false;
    private boolean safetySystemOn = false;

    public void manageEquipment(boolean on) {
        equipmentOn = on;
        System.out.println("Lab equipment " + (on ? "ON" : "OFF"));
    }

    public void manageSafetySystem(boolean on) {
        safetySystemOn = on;
        System.out.println("Lab safety system " + (on ? "ON" : "OFF"));
    }

    public void status() {
        System.out.println("Lab Status: Equipment=" + equipmentOn + ", SafetySystem=" + safetySystemOn);
    }
}

class SmartLibrary implements SmartDevice {
    private int occupancy = 0;
    private int booksAvailable = 100;

    public void trackOccupancy(int people) {
        occupancy = people;
        System.out.println("Library occupancy set to " + occupancy);
```

```
    public void status() {
        System.out.println("Classroom Status: Lights=" + lightsOn + ", AC=" + acOn + ", Projector=" + projectorOn);
    }
}

class SmartLab implements SmartDevice {
    private boolean equipmentOn = false;
    private boolean safetySystemOn = false;

    public void manageEquipment(boolean on) {
        equipmentOn = on;
        System.out.println("Lab equipment " + (on ? "ON" : "OFF"));
    }

    public void manageSafetySystem(boolean on) {
        safetySystemOn = on;
        System.out.println("Lab safety system " + (on ? "ON" : "OFF"));
    }

    public void status() {
        System.out.println("Lab Status: Equipment=" + equipmentOn + ", SafetySystem=" + safetySystemOn);
    }
}

class SmartLibrary implements SmartDevice {
    private int occupancy = 0;
    private int booksAvailable = 100;

    public void trackOccupancy(int people) {
        occupancy = people;
        System.out.println("Library occupancy set to " + occupancy);
```

```
                lab.status();
            } else if (device instanceof SmartLibrary) {
                SmartLibrary library = (SmartLibrary) device;
                library.trackOccupancy(25);
                library.updateBookAvailability(80);
                library.status();
            } else {
                System.out.println("Unknown device type.");
            }
            System.out.println("---");
        }
    }
}
```

OUTPUT→

```
PS C:\Users\Ramesh\Personal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAVA-STEP\Weeks\Week 7\Lab Problems\Program6> cd "c:\Users\Rames
ersonal Folders\MISCELLANEOUS\ENTRANCE EXAMS\SRM\SEMESTERS\SEMESTER-3\JAVA-STEP\Weeks\Week 7\Lab Problems\Program6\" ; if ($?) { javac SmartCampusIOTSyst
ava } ; if ($?) { java SmartCampusIOTSystem }
Classroom lights ON
Classroom AC ON
Classroom projector OFF
Classroom Status: Lights=true, AC=true, Projector=false
---
Lab equipment ON
Lab safety system ON
Lab Status: Equipment=true, SafetySystem=true
---
Library occupancy set to 25
Library books available: 80
Library Status: Occupancy=25, BooksAvailable=80
---
Classroom lights ON
Classroom AC ON
Classroom projector OFF
Classroom Status: Lights=true, AC=true, Projector=false
---
Lab equipment ON
Lab safety system ON
Lab Status: Equipment=true, SafetySystem=true
---
```