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

PROBLEM 1: Food Delivery App (Method Overloading)

```
// File: FoodDeliveryApp.java
public class FoodDeliveryApp {
    public void calculateDelivery(double distance) {
        double cost = distance * 5;
        System.out.println("Basic delivery: Distance " + distance + " km = Rs." + cost);
    }

    public void calculateDelivery(double distance, double priorityFee) {
        double cost = distance * 5 + priorityFee;
        System.out.println("Premium delivery: Distance " + distance + " km + Priority Fee "
+ priorityFee + " = Rs." + cost);
    }

    public void calculateDelivery(double distance, int orders) {
        double discount = orders * 2;
        double cost = distance * 5 - discount;
        System.out.println("Group delivery: Distance " + distance + " km - Discount " +
discount + " = Rs." + cost);
    }

    public void calculateDelivery(double distance, double discountPercent, double
freeLimit) {
        double cost = distance * 5;
        if (cost > freeLimit) {
            System.out.println("Festival special: Delivery free! Order above Rs." + freeLimit);
        } else {
            double discounted = cost - (cost * discountPercent / 100);
            System.out.println("Festival special: Distance " + distance + " km - " +
discountPercent + "% off = Rs." + discounted);
        }
    }

    public static void main(String[] args) {
        FoodDeliveryApp app = new FoodDeliveryApp();
        app.calculateDelivery(10);
        app.calculateDelivery(8, 20);
        app.calculateDelivery(12, 3);
        app.calculateDelivery(15, 20, 100);
    }
}
```

PROBLEM 2: Social Media Feed (Method Overriding)

```
// File: SocialMediaFeed.java
import java.time.LocalDateTime;

class SocialMediaPost {
    protected String author;
    protected String content;
    protected LocalDateTime time;

    public SocialMediaPost(String author, String content) {
        this.author = author;
        this.content = content;
        this.time = LocalDateTime.now();
    }

    public void display() {
        System.out.println(author + " posted: " + content + " at " + time);
    }
}

class InstagramPost extends SocialMediaPost {
    private int likes;
    public InstagramPost(String author, String content, int likes) {
        super(author, content);
        this.likes = likes;
    }
    @Override
    public void display() {
        System.out.println("📸 Instagram by @" + author + ": " + content + " | Likes: " +
likes);
    }
}

class TwitterPost extends SocialMediaPost {
    private int retweets;
    public TwitterPost(String author, String content, int retweets) {
        super(author, content);
        this.retweets = retweets;
    }
    @Override
    public void display() {
        System.out.println("🐦 Tweet by @" + author + ": " + content + " (" +
content.length() + " chars) | Retweets: " + retweets);
    }
}

class LinkedInPost extends SocialMediaPost {
    private int connections;
    public LinkedInPost(String author, String content, int connections) {
        super(author, content);
        this.connections = connections;
    }
    @Override
    public void display() {
        System.out.println("💼 LinkedIn by " + author + ": " + content + " | Connections: " +
connections);
    }
}

public class SocialMediaFeed {
    public static void main(String[] args) {
        SocialMediaPost[] posts = {
            new InstagramPost("alice", "Beach vibes 🌊", 230),
            new TwitterPost("bob", "Java is powerful!", 45),
            new LinkedInPost("carol", "Excited to start a new job!", 500)
        };
        for (SocialMediaPost post : posts) {
            post.display();
        }
    }
}
```

PROBLEM 3: Gaming Character System (Dynamic Method Dispatch)

```
// File: GamingSystem.java
abstract class Character {
    protected String name;
    public Character(String name) {
        this.name = name;
    }
    public abstract void attack();
}

class Warrior extends Character {
    public Warrior(String name) { super(name); }
    @Override
    public void attack() {
        System.out.println(name + " swings a mighty sword! High defense!");
    }
}

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

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

public class GamingSystem {
    public static void main(String[] args) {
        Character[] army = {
            new Warrior("Thor"),
            new Mage("Merlin"),
            new Archer("Robin")
        };
        for (Character c : army) {
            c.attack();
        }
    }
}
```

PROBLEM 4: University Library System (Upcasting)

```
// File: LibrarySystem.java
class LibraryUser {
    protected String name;
    public LibraryUser(String name) {
        this.name = name;
    }
    public void enterLibrary() {
        System.out.println(name + " entered the library.");
    }
}

class Student extends LibraryUser {
    public Student(String name) { super(name); }
    public void borrowBook() {
        System.out.println(name + " borrowed a book.");
    }
    public void useComputer() {
        System.out.println(name + " is using a computer.");
    }
}

class Faculty extends LibraryUser {
    public Faculty(String name) { super(name); }
    public void reserveBook() {
        System.out.println(name + " reserved a book.");
    }
    public void accessDatabase() {
        System.out.println(name + " accessed research database.");
    }
}

class Guest extends LibraryUser {
    public Guest(String name) { super(name); }
    public void browseBooks() {
        System.out.println(name + " is browsing books.");
    }
}

public class LibrarySystem {
    public static void main(String[] args) {
        LibraryUser u1 = new Student("Alice");
        LibraryUser u2 = new Faculty("Dr. Bob");
        LibraryUser u3 = new Guest("Charlie");

        u1.enterLibrary();
        u2.enterLibrary();
        u3.enterLibrary();
    }
}
```

PROBLEM 5: Movie Streaming Platform (Downcasting)

```
// File: StreamingPlatform.java
class Content {
    protected String title;
    public Content(String title) {
        this.title = title;
    }
    public void play() {
        System.out.println("Playing: " + title);
    }
}

class Movie extends Content {
    private int rating;
    public Movie(String title, int rating) {
        super(title);
        this.rating = rating;
    }
    public void showSubtitles() {
        System.out.println("Showing subtitles for " + title);
    }
}

class TVSeries extends Content {
    private int seasons;
    public TVSeries(String title, int seasons) {
        super(title);
        this.seasons = seasons;
    }
    public void nextEpisode() {
        System.out.println("Loading next episode of " + title);
    }
}

class Documentary extends Content {
    private String tag;
    public Documentary(String title, String tag) {
        super(title);
        this.tag = tag;
    }
    public void relatedContent() {
        System.out.println("Showing related documentaries on " + tag);
    }
}

public class StreamingPlatform {
    public static void main(String[] args) {
        Content c = new Movie("Avengers", 5);
        c.play();
        Movie m = (Movie) c;
        m.showSubtitles();

        c = new TVSeries("Stranger Things", 4);
        c.play();
        TVSeries t = (TVSeries) c;
        t.nextEpisode();
    }
}
```

PROBLEM 6: Smart Campus IoT System (Safe Downcasting with `instanceof`)

```
// File: SmartCampus.java
abstract class Device {
   protected String name;
   public Device(String name) { this.name = name; }
   public abstract void status();
}

class SmartClassroom extends Device {
   public SmartClassroom(String name) { super(name); }
   public void controlProjector() {
      System.out.println(name + ": Projector turned on.");
   }
   @Override
   public void status() {
      System.out.println(name + " is a Smart Classroom.");
   }
}

class SmartLab extends Device {
   public SmartLab(String name) { super(name); }
   public void manageEquipment() {
      System.out.println(name + ": Equipment calibrated.");
   }
   @Override
   public void status() {
      System.out.println(name + " is a Smart Lab.");
   }
}

class SmartLibrary extends Device {
   public SmartLibrary(String name) { super(name); }
   public void trackOccupancy() {
      System.out.println(name + ": Occupancy tracked.");
   }
   @Override
   public void status() {
        System.out.println(name + " is a Smart Library.");
    }
}

public class SmartCampus {
    public static void main(String[] args) {
        Device[] devices = {
            new SmartClassroom("Classroom A"),
            new SmartLab("Physics Lab"),
            new SmartLibrary("Central Library")
        };
        for (Device d : devices) {
            d.status();
            if (d instanceof SmartClassroom) {
                ((SmartClassroom) d).controlProjector();
            } else if (d instanceof SmartLab) {
                ((SmartLab) d).manageEquipment();
            } else if (d instanceof SmartLibrary) {
                ((SmartLibrary) d).trackOccupancy();
            }
        }
    }
}
```