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

PRACTICE PROBLEM 1: Gaming Arena - Method Overloading

```
  // File: GameBattle.java
public class GameBattle {

    // Basic attack method
    public void attack(int damage) {
        System.out.println("Basic attack for " + damage + " points!");
    }

    // Overloaded attack method with weapon
    public void attack(int damage, String weapon) {
        System.out.println("Attacking with " + weapon + " for " + damage + "
points!");
    }

    // Overloaded attack method with critical hit
    public void attack(int damage, String weapon, boolean isCritical) {
        if (isCritical) {
            System.out.println("CRITICAL HIT! " + weapon + " deals " + (damage * 2)
+ " points!");
        } else {
            // Call the regular weapon attack
            attack(damage, weapon);
        }
    }

    // Overloaded attack method for team attack
    public void attack(int damage, String[] teammates) {
        int teamSize = teammates.length;
        System.out.print("Team attack with ");
        for (int i = 0; i < teamSize; i++) {
            System.out.print(teammates[i]);
```

```
            if (i < teamSize - 1) {
                System.out.print(", ");
            }
        }
        System.out.println(" for " + (damage * teamSize) + " total damage!");
    }

    // Main method for simulation
    public static void main(String[] args) {
        GameBattle battle = new GameBattle();

        // 1. Basic attack with 50 damage
        battle.attack(50);

        // 2. Sword attack with 75 damage
        battle.attack(75, "Sword");

        // 3. Critical bow attack with 60 damage
        battle.attack(60, "Bow", true);

        // 4. Team attack with {"Alice", "Bob"} for 40 base damage
        String[] teammates = {"Alice", "Bob"};
        battle.attack(40, teammates);
    }
}
```

PRACTICE PROBLEM 2: Social Media Platform - Method Overriding

```
  // File: SocialMediaDemo.java
public class SocialMediaPost {
    protected String content;
    protected String author;
    public SocialMediaPost(String content, String author) {
        this.content = content;
        this.author = author;
    }
    public void share() {
```

```
        System.out.println("Sharing: " + content + " by " + author);
    }
}

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

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

class SocialMediaDemo {
    public static void main(String[] args) {
        SocialMediaPost[] feed = new SocialMediaPost[3];
        feed[0] = new InstagramPost("Sunset vibes!", "john_doe", 245);
        feed[1] = new TwitterPost("Java is awesome!", "code_ninja", 89);
        feed[2] = new SocialMediaPost("Hello world!", "beginner");
        for (SocialMediaPost post : feed) {
```

```
            post.share();
        }
    }
}
```

PRACTICE PROBLEM 3: Food Delivery App - Dynamic Method Dispatch

```
  // File: FoodDelivery.java
public class Restaurant {
    protected String name;
    public Restaurant(String name) {
        this.name = name;
    }
    public void prepareFood() {
        System.out.println(name + " is preparing generic food");
    }
    public void estimateTime() {
        System.out.println("Estimated time: 30 minutes");
    }
}

class PizzaPlace extends Restaurant {
    public PizzaPlace(String name) {
        super(name);
    }
    @Override
    public void prepareFood() {
        System.out.println(name + " is making delicious pizza with fresh toppings!");
    }
    @Override
    public void estimateTime() {
        System.out.println("Pizza ready in 20 minutes!");
    }
}

class SushiBar extends Restaurant {
    public SushiBar(String name) {
```

```
        super(name);
    }
    @Override
    public void prepareFood() {
        System.out.println(name + " is crafting fresh sushi with precision!");
    }
    @Override
    public void estimateTime() {
        System.out.println("Sushi will be ready in 25 minutes!");
    }
}

class FoodDelivery {
    public static void main(String[] args) {
        Restaurant r;
        r = new PizzaPlace("Mario's Pizza");
        r.prepareFood();
        r.estimateTime();
        r = new SushiBar("Tokyo Sushi");
        r.prepareFood();
        r.estimateTime();
    }
}
```

PRACTICE PROBLEM 4: University System - Upcasting

```
  // File: UniversitySystem.java
public class Person {
    protected String name;
    protected int age;
    protected String email;
    public Person(String name, int age, String email) {
        this.name = name;
        this.age = age;
        this.email = email;
    }
```

```
    public void introduce() {
        System.out.println("Hi! I'm " + name + ", " + age + " years old.");
    }
    public void getContactInfo() {
        System.out.println("Email: " + email);
    }
}

class Student extends Person {
    private String studentId;
    private String major;
    public Student(String name, int age, String email, String studentId, String
major) {
        super(name, age, email);
        this.studentId = studentId;
        this.major = major;
    }
    public void attendLecture() {
        System.out.println(name + " is attending " + major + " lecture");
    }
    public void submitAssignment() {
        System.out.println("Assignment submitted by " + studentId);
    }
}

class Professor extends Person {
    private String department;
    public Professor(String name, int age, String email, String department) {
        super(name, age, email);
        this.department = department;
    }
    public void conductClass() {
        System.out.println("Prof. " + name + " is teaching " + department);
    }
}

class UniversitySystem {
```

```
    public static void main(String[] args) {
        Person p = new Student("Alice", 20, "alice@uni.edu", "CS2021", "Computer
Science");
        p.introduce();
        p.getContactInfo();
        System.out.println("Accessing field name: " + p.name);
    }
}
```

PRACTICE PROBLEM 5: Entertainment System - Downcasting

```
 // File: EntertainmentHub.java
public class Entertainment {
    protected String title;
    public Entertainment(String title) {
        this.title = title;
    }
    public void start() {
        System.out.println("Starting " + title);
    }
    public void stop() {
        System.out.println("Stopping " + title);
    }
}

class Movie extends Entertainment {
    private String genre;
    public Movie(String title, String genre) {
        super(title);
        this.genre = genre;
    }
    public void showSubtitles() {
        System.out.println("Showing subtitles for " + title + " (" + genre + ")");
    }
    public void adjustQuality() {
        System.out.println("Adjusting video quality for " + title);
```

```
    }
}

class Game extends Entertainment {
    private String platform;
    public Game(String title, String platform) {
        super(title);
        this.platform = platform;
    }
    public void saveProgress() {
        System.out.println("Saving " + title + " progress on " + platform);
    }
    public void showLeaderboard() {
        System.out.println(title + " leaderboard on " + platform);
    }
}

class EntertainmentHub {
    public static void main(String[] args) {
        Entertainment e = new Movie("Avengers", "Action");
        e.start();
        Movie m = (Movie) e;
        m.showSubtitles();
        m.adjustQuality();
        e = new Game("FIFA 24", "PlayStation");
        e.start();
        Game g = (Game) e;
        g.saveProgress();
        g.showLeaderboard();
        // Wrong downcast example:
        // Movie wrong = (Movie) e; // Causes ClassCastException at runtime
    }
}
```