🟢 Bài 1: Quản lý thông tin sinh viên (Dễ - Encapsulation)

Mục tiêu: Ôn tập kỹ thuật đóng gói và sử dụng getter/setter.

Yêu cầu:

- Tạo lớp `Student` với các thuộc tính: `id`, `name`, `age`, `gpa`.
- Tất cả các thuộc tính để ở `private`, cung cấp `getter` và `setter`.
- Viết hàm để:
    - Nhập danh sách n sinh viên.
    - Hiển thị danh sách sinh viên có GPA >= 3.2.
    - Sắp xếp sinh viên theo GPA giảm dần.

Gợi ý:

```
public class Student {
    private String id;
    private String name;
    private int age;
    private double gpa;

    // constructor, getters, setters
}
```

🟡 Bài 2: Hệ thống quản lý nhân sự (Trung bình - Inheritance + Polymorphism)

Mục tiêu: Thực hành kế thừa và đa hình.

Yêu cầu:

- Tạo lớp trừu tượng `Employee` với các thuộc tính: `id`, `name`, `salary`. Có phương thức trừu tượng `calculateSalary()`.
- Tạo 2 lớp kế thừa:
    - `FullTimeEmployee`: lương cố định.

    - `PartTimeEmployee`: tính lương theo số giờ làm.

- Trong hàm `main`, lưu danh sách `Employee` (có thể là cả FullTime và PartTime), hiển thị thông tin và tính tổng lương công ty phải trả.

Gợi ý:

```
abstract class Employee {
    protected String id;
    protected String name;
    public abstract double calculateSalary();
}
```

🔴 Bài 3: Hệ thống quản lý động vật (Khó - Tích hợp đầy đủ 4 đặc trưng OOP)

Mục tiêu: Tổng hợp kiến thức về trừu tượng, kế thừa, đa hình, đóng gói.

Yêu cầu:

- Tạo lớp trừu tượng `Animal` với các phương thức trừu tượng `makeSound()` và `move()`.
- Tạo các lớp con `Dog`, `Cat`, `Bird` kế thừa từ `Animal`.
- Mỗi loài có cách `makeSound()` và `move()` khác nhau.
- Tạo thêm lớp `Zoo` chứa danh sách các `Animal`. Viết phương thức để:
    - Thêm một con vật.
    - In ra tiếng kêu và hành động di chuyển của từng con vật.
    - Thống kê số lượng mỗi loài.

Gợi ý:

```
abstract class Animal {
    private String name;
    public abstract void makeSound();
    public abstract void move();
}
```

🔴 Bài 4: Mô phỏng hệ thống ngân hàng (Advanced OOP Design)

🎯 Mục tiêu:

Sử dụng: `Abstraction`, `Encapsulation`, `Inheritance`, `Polymorphism` để mô phỏng một hệ thống ngân hàng đơn giản.

✅ Yêu cầu:

1. Tạo **lớp trừu tượng** `BankAccount` có:
    - Thuộc tính: `accountNumber`, `ownerName`, `balance`.
    - Phương thức: `deposit()`, `withdraw()`, `calculateInterest()` (trừu tượng).
2. Tạo các lớp con:
    - `SavingsAccount`: lãi suất cố định.
    - `CheckingAccount`: không có lãi suất, nhưng có hạn mức rút tiền.
3. Thêm lớp `Bank` quản lý danh sách tài khoản:
    - Thêm tài khoản mới.
    - Rút tiền, gửi tiền theo số tài khoản.
    - Tính tổng số tiền trong hệ thống.
    - Ghi log giao dịch.
4. Áp dụng đa hình khi gọi `calculateInterest()` cho tất cả tài khoản.

💡 Gợi ý mở rộng:

- Sử dụng `interface Loggable` cho các lớp cần log hoạt động.
- Cân nhắc dùng `HashMap<String, BankAccount>` để quản lý tài khoản theo `accountNumber`.

🔴 Bài 5: Quản lý hệ thống học tập trực tuyến (Mini LMS - Learning Management System)

🎯 Mục tiêu:

Xây dựng một hệ thống mô phỏng ứng dụng học online như Udemy/Coursera.

✅ Yêu cầu:

1. **Lớp trừu tượng `User`**:
    - Thuộc tính: `id`, `name`, `email`.
    - Phương thức: `login()`, `logout()`.
2. **Lớp kế thừa**:
    - `Instructor`: có thêm `courseList`, phương thức `createCourse()`, `gradeStudent()`.
    - `Student`: có thêm `enrolledCourses`, phương thức `enrollCourse()`, `viewGrades()`.
3. **Lớp `Course`**:
    - Thuộc tính: `courseId`, `title`, `instructor`, `List<Student> students`, `Map<Student, Double> grades`.
4. Hệ thống có lớp `LMS`:
    - Thêm user mới, tạo khóa học, đăng ký khóa học, chấm điểm, in bảng điểm.
    - Cho phép nhiều học sinh ghi danh vào một khóa học.
    - Đảm bảo dùng **đa hình** khi gọi các phương thức login/logout.

💡 Gợi ý mở rộng:

- Tạo interface `Gradable` cho các thực thể có khả năng chấm điểm.

- Lưu trữ dữ liệu bằng file hoặc serialization (tùy bạn muốn mở rộng).