CODEFORGE - BUỔI 11: CLASSES & OBJECTS

25 BÀI TẬP (18 CORE + 7 ADVANCED)

Kiến thức Buổi 11:

- Classes & Objects (class declaration, fields, methods, object creation, references, null)
- Constructors (default, parameterized, overloading, this, chaining, copy constructor)
- Access Modifiers (private, public, getters/setters, encapsulation)

CodeForge - B01 - Class Với Fields Cơ Bản

Độ khó: ★ Easy
Loại: Competitive

Đề bài

Tạo class `Student` với các fields:

```
class Student {
    String name;
    int age;
    double gpa;
}
```

Nhập thông tin 1 sinh viên, in ra màn hình.

◇ Input

```
<name>
<age>
<gpa>
```

◇ Output

```
Thông tin sinh viên:
Tên: <name>
Tuổi: <age>
Điểm TB: <gpa>
```

Ví dụ

Test case 1

Input:

```
Alice
20
3.5
```

Output:

```
Thông tin sinh viên:
Tên: Alice
Tuổi: 20
Điểm TB: 3.5
```

Tags: oop, class, fields, competitive

CodeForge - B02 - Methods Tính Toán

Độ khó: ★ Easy
Loại: Competitive

Đề bài

Class `Rectangle` với methods tính diện tích và chu vi:

```
class Rectangle {
    double width;
    double height;

    double getArea() {
        return width * height;
    }

    double getPerimeter() {
        return 2 * (width + height);
    }
}
```

Input

```
<width>
<height>
```

Output

```
Diện tích: <area>
Chu vi: <perimeter>
```

Ví dụ

Test case 1

Input:

```
5.0
10.0
```

Output:

```
Diện tích: 50.0
Chu vi: 30.0
```

Tags: oop, methods, competitive

CodeForge - B03 - Methods Có Tham Số

Độ khó: ★★ Medium
Loại: Competitive

Đề bài

Class `BankAccount` với deposit/withdraw:

```
class BankAccount {
    double balance;

    void deposit(double amount) {
        balance += amount;
    }

    void withdraw(double amount) {
        if (amount <= balance) {
            balance -= amount;
        }
    }
}
```

Thực hiện N giao dịch.

Input

- Dòng 1: balance (số dư ban đầu)
- Dòng 2: N (số giao dịch)
- N dòng: D amount (Deposit) hoặc W amount (Withdraw)

Output

- Số dư sau mỗi giao dịch

Ví dụ

Test case 1

Input:

```
1000.0
5
D 500.0
W 200.0
D 300.0
```

```
W 2000.0
W 100.0
```

Output:

```
Số dư ban đầu: 1000.0đ
GD 1: Nộp 500.0đ → Số dư: 1500.0đ
GD 2: Rút 200.0đ → Số dư: 1300.0đ
GD 3: Nộp 300.0đ → Số dư: 1600.0đ
GD 4: Rút 2000.0đ → Không đủ! Số dư: 1600.0đ
GD 5: Rút 100.0đ → Số dư: 1500.0đ
```

Tags: oop, methods, parameters, competitive

CodeForge - B04 - Constructor Mặc Định

Độ khó: ★ Easy
Loại: Competitive

Đề bài

Class Book với default constructor:

```
class Book {
    String title;
    int pages;

    Book() {
        title = "Unknown";
        pages = 0;
    }
}
```

◇ Input

```
<choice>
```

- 1: Dùng default constructor
- 2: Tạo rồi gán giá trị sau

◇ Output

- Thông tin sách

Ví dụ

Test case 1

Input:

```
1
```

Output:

```
Sách (default constructor):
Tiêu đề: Unknown
```

```
Số trang: 0
```

Test case 2

Input:

```
2
Java-Programming
500
```

Output:

```
Sách (gán sau):
Tiêu đề: Java Programming
Số trang: 500
```

Tags: oop, constructor, default, competitive

CodeForge - B05 - Constructor Có Tham Số

Độ khó: ★★ Medium
Loại: Competitive

Đề bài

Class `Product` với parameterized constructor:

```
class Product {
    String name;
    double price;
    int quantity;

    Product(String n, double p, int q) {
        name = n;
        price = p;
        quantity = q;
    }

    double getTotalValue() {
        return price * quantity;
    }
}
```

Nhập N sản phẩm, tính tổng giá trị kho.

Input

- Dòng 1: N
- N nhóm 3 dòng: name, price, quantity

Output

- Thông tin từng sản phẩm và tổng giá trị

Ví dụ

Test case 1

Input:

```
3
Laptop
20000000
5
Mouse
```

```
500000
20
Keyboard
1500000
10
```

Output:

```
Sản phẩm 1: Laptop
Giá: 20000000đ × 5 = 100000000đ

Sản phẩm 2: Mouse
Giá: 500000đ × 20 = 10000000đ

Sản phẩm 3: Keyboard
Giá: 1500000đ × 10 = 15000000đ

Tổng giá trị kho: 125000000đ
```

Tags: oop, constructor, parameterized, competitive

CodeForge - B06 - Constructor Overloading

Độ khó: ★★ Medium
Loại: Competitive

Đề bài

Class `Circle` với nhiều constructors:

```
class Circle {
    double radius;

    Circle() {
        radius = 1.0;
    }

    Circle(double r) {
        radius = r;
    }

    double getArea() {
        return 3.14159 * radius * radius;
    }
}
```

Input

- Dòng 1: N (số circles)
- N dòng: 0 (default) hoặc

Output

- Thông tin từng circle

Ví dụ

Test case 1

Input:

```
4
0
5.0
10.0
0
```

Output:

```
Circle 1: r = 1.0 (default), diện tích = 3.14
Circle 2: r = 5.0, diện tích = 78.54
Circle 3: r = 10.0, diện tích = 314.16
Circle 4: r = 1.0 (default), diện tích = 3.14
```

Tags: oop, constructor, overloading, competitive

CodeForge - B07 - Từ Khóa this

Độ khó: ★★ Medium
Loại: Competitive

Đề bài

Class `Employee` sử dụng `this`:

```
class Employee {
    String name;
    double salary;

    Employee(String name, double salary) {
        this.name = name;  // this phân biệt field vs parameter
        this.salary = salary;
    }

    void display() {
        System.out.println("Tên: " + this.name);
        System.out.println("Lương: " + this.salary + "đ");
    }
}
```

Input

```
<name>
<salary>
```

Output

- Giải thích vai trò `this` và thông tin nhân viên

Ví dụ

Test case 1

Input:

```
Alice
15000000
```

Output:

```
Constructor:
- Parameter name: Alice
- this.name: Gán vào field name của object

Nhân viên:
Tên: Alice
Lương: 15000000đ

Vai trò this:
✓ Phân biệt field (this.name) vs parameter (name)
✓ Tham chiếu đến object hiện tại
```

Tags: oop, this, keyword, competitive

CodeForge - B08 - Constructor Chaining (this)

Độ khó: ★ ★ ★ Hard
Loại: Competitive

Đề bài

Class `Car` với constructor chaining:

```
class Car {
    String brand;
    String model;
    int year;

    Car() {
        this("Unknown", "Unknown", 2024);
    }

    Car(String brand) {
        this(brand, "Unknown", 2024);
    }

    Car(String brand, String model) {
        this(brand, model, 2024);
    }

    Car(String brand, String model, int year) {
        this.brand = brand;
        this.model = model;
        this.year = year;
    }
}
```

◇ Input

- Dòng 1: N (số xe)
- N dòng:
  - 0: Car()
  - 1 brand: Car(brand)
  - 2 brand model: Car(brand, model)
  - 3 brand model year: Car(brand, model, year)

◇ Output

- Thông tin từng xe

Ví dụ

Test case 1

Input:

```
4
0
1 Toyota
2 Honda Civic
3 BMW X5 2023
```

Output:

```
Xe 1: Unknown / Unknown / 2024 (0 tham số)
Xe 2: Toyota / Unknown / 2024 (1 tham số)
Xe 3: Honda / Civic / 2024 (2 tham số)
Xe 4: BMW / X5 / 2023 (3 tham số)

Constructor chaining:
✓ this() gọi constructor khác trong cùng class
✓ Tránh lặp code khởi tạo
```

Tags: oop, constructor, chaining, this, competitive

CodeForge - B09 - Copy Constructor

Độ khó: ★★ Medium
Loại: Competitive

Đề bài

Class `Point` với copy constructor:

```
class Point {
    int x;
    int y;

    Point(int x, int y) {
        this.x = x;
        this.y = y;
    }

    Point(Point other) {  // Copy constructor
        this.x = other.x;
        this.y = other.y;
    }
}
```

Input

```
<x> <y>
```

Output

- Demo copy constructor

Ví dụ

Test case 1

Input:

```
5 10
```

Output:

```
Point gốc p1: (5, 10)

Copy constructor:
Point p2 = new Point(p1)

Point p2: (5, 10)

Thay đổi p2 thành (20, 30)

Kết quả:
p1: (5, 10)  ← Không đổi
p2: (20, 30)

Copy constructor:
✓ Tạo object MỚI với giá trị giống
✓ 2 objects độc lập
```

Tags: oop, constructor, copy, competitive

CodeForge - B10 - Reference Variables

Độ khó: ★ ★ Medium
Loại: Competitive

Đề bài

Demo references và objects:

```
class Student {
    String name;
    int age;
}

Student s1 = new Student();
s1.name = "Alice";
s1.age = 20;

Student s2 = s1;  // s2 trỏ cùng object với s1
s2.age = 25;      // Thay đổi qua s2 → ảnh hưởng s1
```

Input

```
<name> <age>
<newAge>
```

Output

- Demo reference behavior

Ví dụ

Test case 1

Input:

```
Alice 20
25
```

Output:

```
Tạo s1: Alice, 20 tuổi

Student s2 = s1 (COPY reference)

Thay đổi s2.age = 25

Kết quả:
s1: Alice, 25 tuổi  ← Thay đổi!
s2: Alice, 25 tuổi

Giải thích:
✓ s1 và s2 trỏ đến CÙNG 1 object
✓ Thay đổi qua s2 → ảnh hưởng s1
```

Tags: oop, reference, object, competitive

CodeForge - B11 - Null Reference

Độ khó: ★ ★ Medium
Loại: Competitive

Đề bài

Demo null reference:

```
class Student {
    String name;
    int age;
}

Student s = null;  // s không trỏ đến object nào
s.name = "Alice";  // NullPointerException!
```

Input

```
<choice>
```

- 1: Tạo object bình thường
- 2: Dùng null reference

Output

- Demo null behavior

Ví dụ

Test case 1

Input:

```
1
Alice 20
```

Output:

```
Student s = new Student();
s.name = "Alice";
s.age = 20;
```

```
Thành công!
Thông tin: Alice, 20 tuổi
```

Test case 2

Input:

```
2
```

Output:

```
Student s = null;
s KHÔNG trỏ đến object nào!

Cố gắng: s.name = "Alice"
❌ NullPointerException!

Cách tránh:
✓ Kiểm tra: if (s != null)
✓ Khởi tạo: s = new Student()
```

Tags: oop, null, reference, competitive

CodeForge - B12 - Mảng Objects

Độ khó: ★ ★ ★ Hard
Loại: Competitive

Đề bài

Tạo mảng objects và tìm max:

```
class Student {
    String name;
    double gpa;

    Student(String name, double gpa) {
        this.name = name;
        this.gpa = gpa;
    }
}

Student[] students = new Student[N];
```

Tìm sinh viên có GPA cao nhất.

Input

- Dòng 1: N
- N nhóm 2 dòng: name, gpa

Output

- Sinh viên có GPA cao nhất

Ví dụ

Test case 1

Input:

```
5
Alice
3.5
Bob
3.8
Charlie
3.2
David
3.9
```

```
Eve
3.6
```

Output:

```
Danh sách (5 sinh viên):
1. Alice - GPA: 3.5
2. Bob - GPA: 3.8
3. Charlie - GPA: 3.2
4. David - GPA: 3.9
5. Eve - GPA: 3.6

Sinh viên GPA cao nhất:
Tên: David
GPA: 3.9
```

Tags: oop, array, objects, search, competitive

CodeForge - B13 - Private Fields & Getters

Độ khó: ★★ Medium
Loại: Competitive

Đề bài

Class với private fields và getters:

```
class BankAccount {
    private double balance;  // private - ẩn bên ngoài

    BankAccount(double initialBalance) {
        balance = initialBalance;
    }

    double getBalance() {  // getter - truy cập balance
        return balance;
    }
}
```

Input

```
<initialBalance>
```

Output

- Demo private và getter

Ví dụ

Test case 1

Input:

```
5000000
```

Output:

```
Tạo tài khoản: 5000000đ

private double balance:
```

```
❌ acc.balance → Không truy cập được (private)

double getBalance():
✓ acc.getBalance() → 5000000đ (OK)

Lợi ích private:
✓ Ẩn dữ liệu bên ngoài class
✓ Kiểm soát truy cập qua methods
```

Tags: oop, private, getter, encapsulation, competitive

CodeForge - B14 - Setters Với Validation

Độ khó: ★★ Medium
Loại: Competitive

Đề bài

Class với setter có validation:

```
class Product {
    private double price;

    void setPrice(double p) {
        if (p > 0) {
            price = p;
        } else {
            System.out.println("Giá phải > 0!");
        }
    }

    double getPrice() {
        return price;
    }
}
```

Input

- Dòng 1: N (số lần set price)
- N dòng: price

Output

- Kết quả từng lần set

Ví dụ

Test case 1

Input:

```
4
50000
-10000
0
100000
```

Output:

```
Set 1: setPrice(50000)
✓ Hợp lệ → Price = 50000đ

Set 2: setPrice(-10000)
❌ Giá phải > 0!
Price = 50000đ (không đổi)

Set 3: setPrice(0)
❌ Giá phải > 0!
Price = 50000đ (không đổi)

Set 4: setPrice(100000)
✓ Hợp lệ → Price = 100000đ

Lợi ích setter:
✓ Kiểm tra dữ liệu hợp lệ
✓ Tránh giá trị không đúng
```

Tags: oop, setter, validation, encapsulation, competitive

CodeForge - B15 - Encapsulation Hoàn Chỉnh

Độ khó: ★ ★ ★ Hard
Loại: Competitive

Đề bài

Class với encapsulation đầy đủ:

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

    Student(String name, int age, double gpa) {
        setName(name);
        setAge(age);
        setGpa(gpa);
    }

    void setName(String n) {
        if (n != null && !n.isEmpty()) {
            name = n;
        }
    }

    void setAge(int a) {
        if (a > 0 && a < 100) {
            age = a;
        }
    }

    void setGpa(double g) {
        if (g >= 0.0 && g <= 4.0) {
            gpa = g;
        }
    }

    String getName() { return name; }
    int getAge() { return age; }
    double getGpa() { return gpa; }
}
```

Input

- Dòng 1: name, age, gpa (hợp lệ)
- Dòng 2-4: Các lần update (có cả không hợp lệ)

◇ Output

- Kết quả update

📊 Ví dụ

Test case 1

Input:

```
Alice 20 3.5
Bob
-5
5.0
```

Output:

```
Tạo sinh viên: Alice, 20 tuổi, GPA: 3.5

Update 1: setName("Bob")
✓ Hợp lệ → Name = Bob

Update 2: setAge(-5)
❌ Tuổi không hợp lệ! (Age = 20)

Update 3: setGpa(5.0)
❌ GPA không hợp lệ! (GPA = 3.5)

Thông tin cuối:
Tên: Bob
Tuổi: 20
GPA: 3.5

Encapsulation:
✓ private fields - ẩn dữ liệu
✓ getters - truy cập an toàn
✓ setters - validation tự động
```

Tags: oop, encapsulation, getters, setters, validation, competitive

CodeForge - B16 - So Sánh Objects

Độ khó: ★ ★ ★ Hard
Loại: Competitive

Đề bài

Method so sánh 2 objects:

```
class Point {
    private int x;
    private int y;

    Point(int x, int y) {
        this.x = x;
        this.y = y;
    }

    boolean isEqual(Point other) {
        return this.x == other.x && this.y == other.y;
    }

    int getX() { return x; }
    int getY() { return y; }
}
```

Input

```
<x1> <y1>
<x2> <y2>
```

Output

- Kết quả so sánh

Ví dụ

Test case 1

Input:

```
5 10
5 10
```

Output:

```
Point p1: (5, 10)
Point p2: (5, 10)

So sánh reference:
p1 == p2: false  ← Khác reference

So sánh nội dung:
p1.isEqual(p2): true  ← Cùng giá trị x, y
```

Test case 2

Input:

```
5 10
8 12
```

Output:

```
Point p1: (5, 10)
Point p2: (8, 12)

So sánh nội dung:
p1.isEqual(p2): false  ← Khác giá trị
```

Tags: oop, comparison, equals, competitive

CodeForge - B17 - Tìm Kiếm Trong Mảng Objects

Độ khó: ★ ★ ★ Hard
Loại: Competitive

Đề bài

Tìm kiếm trong mảng objects:

```
class Product {
    private String id;
    private String name;
    private double price;

    // Constructor, getters

    boolean hasId(String searchId) {
        return id.equals(searchId);
    }
}
```

Tìm sản phẩm theo ID.

Input

- Dòng 1: N
- N nhóm 3 dòng: id, name, price
- Dòng cuối: searchId

Output

- Kết quả tìm kiếm

Ví dụ

Test case 1

Input:

```
3
P001
Laptop
20000000
P002
Mouse
500000
```

```
P003
Keyboard
1500000
P002
```

Output:

```
Kho hàng (3 sản phẩm):
1. P001 - Laptop: 20000000đ
2. P002 - Mouse: 500000đ
3. P003 - Keyboard: 1500000đ

Tìm sản phẩm ID: P002

Tìm thấy:
Mã: P002
Tên: Mouse
Giá: 500000đ
```

Tags: oop, search, array, objects, competitive

CodeForge - B18 - Quản Lý Mảng Objects

Độ khó: ★ ★ ★ Hard
Loại: Competitive

Đề bài

Quản lý mảng sinh viên với các operations:

```
class Student {
    private String id;
    private String name;
    private double gpa;

    // Constructor, getters, setters
}

Student[] students = new Student[100];
int count = 0;  // Số lượng hiện tại
```

Commands: ADD, FIND, UPDATE, DELETE, LIST, COUNT

Input

- Dòng 1: N (commands)
- N dòng: Commands

Output

- Kết quả từng command

Ví dụ

Test case 1

Input:

```
8
ADD SV001 Alice 3.5
ADD SV002 Bob 3.8
FIND SV001
UPDATE SV002 3.9
DELETE SV003
DELETE SV001
LIST
COUNT
```

Output:

```
[1] ADD: Đã thêm SV001 - Alice (GPA: 3.5)
[2] ADD: Đã thêm SV002 - Bob (GPA: 3.8)
[3] FIND SV001: Alice (GPA: 3.5)
[4] UPDATE SV002: GPA 3.8 → 3.9
[5] DELETE SV003: Không tìm thấy
[6] DELETE SV001: Đã xóa Alice
[7] LIST (1 sinh viên):
    1. SV002 - Bob (GPA: 3.9)
[8] COUNT: 1 sinh viên
```

Tags: oop, array, crud, management, competitive

CodeForge - B19D - Thiết Kế Class Rectangle

Độ khó: ★★ Medium
Loại: Design

Đề bài

Thiết kế class `Rectangle` với encapsulation đầy đủ:

Yêu cầu:

- Private fields: width, height
- Constructor với validation (width, height > 0)
- Getters và setters với validation
- Methods:
  - `double getArea()`
  - `double getPerimeter()`
  - `boolean isSquare()`
  - `void scale(double factor)` - nhân kích thước

Không có input/output test cases - Focus vào thiết kế class.

Nhiệm vụ

Viết code hoàn chỉnh và demo:

1. Tạo Rectangle(5, 10)
2. In area, perimeter
3. Check isSquare()
4. Scale × 2
5. In area, perimeter mới

Ví dụ Demo

```
Rectangle r = new Rectangle(5, 10);
System.out.println("Area: " + r.getArea());        // 50.0
System.out.println("Perimeter: " + r.getPerimeter()); // 30.0
System.out.println("Is square: " + r.isSquare());     // false

r.scale(2);
System.out.println("Area after scale: " + r.getArea()); // 200.0
```

Tags: `oop`, `class-design`, `encapsulation`, `design`

CodeForge - B20D - Thiết Kế Class Circle

Độ khó: ★ ★ Medium
Loại: Design

Đề bài

Thiết kế class `Circle`:

Yêu cầu:

- Private field: radius
- Constructor với validation (radius > 0)
- Getter và setter với validation
- Methods:
  - `double getArea()` - $\pi \times r^2$
  - `double getCircumference()` - $2 \times \pi \times r$
  - `double getDiameter()` - $2 \times r$
  - `boolean fitsIn(Circle other)` - có nằm trong circle khác?

Nhiệm vụ

Implement và demo:

1. Tạo Circle(5)
2. In area, circumference, diameter
3. Tạo Circle(10)
4. Check circle1.fitsIn(circle2)

Tags: `oop`, `class-design`, `encapsulation`, `design`

CodeForge - B21D - Thiết Kế Class BankAccount

Độ khó: ★ ★ ★ Hard
Loại: Design

Đề bài

Thiết kế class `BankAccount` với encapsulation và business logic:

Yêu cầu:

- Private fields: accountNumber, ownerName, balance
- Constructor khởi tạo balance >= 0
- Methods:
  - `void deposit(double amount)` - amount > 0
  - `boolean withdraw(double amount)` - kiểm tra đủ tiền
  - `boolean transfer(BankAccount target, double amount)`
  - `double getBalance()`
- **Không có setter cho balance** - chỉ thay đổi qua methods

Focus: Encapsulation và logic validation.

Nhiệm vụ

Implement và demo transfer giữa 2 accounts.

Tags: `oop`, `class-design`, `encapsulation`, `business-logic`, `design`

CodeForge - B22D - Thiết Kế Class Date

Độ khó: ★ ★ ★ Hard
Loại: Design

Đề bài

Thiết kế class `Date` (không dùng java.time):

Yêu cầu:

- Private fields: day, month, year
- Constructor với validation phức tạp:
  - 1 ≤ day ≤ daysInMonth(month, year)
  - 1 ≤ month ≤ 12
  - year > 0
- Methods:
  - `boolean isLeapYear()`
  - `int daysInMonth()`
  - `void nextDay()` - chuyển sang ngày tiếp theo
  - `int compareTo(Date other)` - so sánh
  - `String toString()` - format "dd/mm/yyyy"

Focus: Logic validation phức tạp, năm nhuận, cuối tháng.

Nhiệm vụ

Implement và test:

- Date(28, 2, 2024) - leap year
- nextDay() qua tháng 3
- compareTo

Tags: `oop`, `class-design`, `validation`, `design`

CodeForge - B23D - Thiết Kế Class Fraction

Độ khó: ★ ★ ★ Hard
Loại: Design

Đề bài

Thiết kế class `Fraction` (phân số):

Yêu cầu:

- Private fields: numerator (tử), denominator (mẫu)
- Constructor:
  - Validation: denominator ≠ 0
  - Tự động rút gọn (GCD)
- Methods:
  - `Fraction add(Fraction other)` - trả về phân số mới
  - `Fraction subtract(Fraction other)`
  - `Fraction multiply(Fraction other)`
  - `Fraction divide(Fraction other)`
  - `void simplify()` - rút gọn
  - `String toString()` - format "a/b"

Focus: Immutability (methods trả về object mới, không sửa this).

Nhiệm vụ

Implement và demo: 1/2 + 1/3 = 5/6

Tags: `oop`, `class-design`, `immutability`, `design`

CodeForge - B24D - Thiết Kế Class Point & Line

Độ khó: ★ ★ ★ Hard
Loại: Design

Đề bài

Thiết kế 2 classes có quan hệ:

Class Point:

- Private fields: x, y
- Methods:
  - `double distanceTo(Point other)` - khoảng cách
  - `boolean isEqual(Point other)`

Class Line:

- Private fields: Point start, Point end
- Methods:
  - `double getLength()` - độ dài
  - `Point getMidpoint()` - trung điểm
  - `boolean isParallel(Line other)` - song song?

Focus: Composition (Line chứa 2 Points).

Nhiệm vụ

Implement và demo line operations.

Tags: `oop`, `class-design`, `composition`, `design`

CodeForge - B25D - Thiết Kế Mini Library System

Độ khó: ★ ★ ★ Hard
Loại: Design (CAPSTONE)

Đề bài

CAPSTONE DESIGN - Thiết kế hệ thống thư viện mini với 3 classes:

Class Book:

- Private: isbn, title, author, available
- Methods: borrow(), return(), display()

Class Member:

- Private: id, name, borrowedBooks[] (mảng String ISBN)
- Methods: canBorrow(), addBook(), removeBook()

Class Library:

- Private: books[] (mảng Book), members[] (mảng Member)
- Methods: addBook(), addMember(), borrowBook(), returnBook()

Yêu cầu:

- Encapsulation đầy đủ (private fields, getters/setters)
- Validation (member chỉ mượn tối đa 3 cuốn)
- Relationships giữa classes

Nhiệm vụ

Implement 3 classes và demo workflow:

1. Thêm books và members
2. Mượn sách
3. Trả sách
4. Check giới hạn

Tags: `oop`, `class-design`, `capstone`, `encapsulation`, `composition`, `design`