BUỔI 5: THỰC HÀNH BÀI TẬP LỚP & ĐỐI TƯỢNG

Thời lượng: 3 tiết

1 Mục tiêu buổi học

- Ôn tập các khái niệm chính về lớp và đối tượng trong Java.
- Vận dụng các kiến thức về **constructor, overloading, encapsulation, phạm vi truy cập, this, static, package, composition.**
- Giải quyết bài tập lập trình hướng đối tượng thực tế.
- Nâng cao kỹ năng lập trình thông qua thực hành có hướng dẫn.

2 Nội dung buổi học

🔷 Phần 1: Ôn tập nhanh (30 phút)

- **Lớp & Đối tượng**: Khái niệm, cách khai báo, sử dụng.
- **Constructor & Overloading**: Cách khởi tạo đối tượng với nhiều kiểu tham số.
- **Phạm vi truy cập (private, protected, public, default)**: Ứng dụng trong lập trình. **Từ khóa this, static**: Khi nào cần dùng?
- **Composition (Quan hệ HAS-A)**: Xây dựng quan hệ giữa các lớp.

📌 **Hoạt động**: Câu hỏi trắc nghiệm nhanh (5 phút).

🔷 Phần 2: Hướng dẫn thực hành (45 phút)

✔️ *Bài tập 1: Xây dựng lớp mô phỏng một hệ thống quản lý sản phẩm*

- Tạo lớp Productvới thuộc tính id, name, price, quantityvà phương thức phù hợp.
- Viết chương trình quản lý danh sách sản phẩm, thực hiện thêm, xóa, tìm kiếm.

  Mở rộng: Sử dụng constructor để tạo sản phẩm nhanh chóng.

✔️ *Bài tập 2: Xây dựng lớp kế thừa & đa hình*

- Tạo lớp Vehicle(phương tiện) và các lớp con Car, Motorbikevới phương thức chung và riêng.
- Sử dụng **overriding** để ghi đè phương thức.
- Viết chương trình quản lý danh sách phương tiện và hiển thị thông tin.

📌 **Lưu ý**: Áp dụng **phạm vi truy cập**, **getter/setter**, **constructor** để đảm bảo tính đóng gói.

🔷 **Phần 3: Bài tập nâng cao (45 phút)**

✔️ *Bài tập 3: Xây dựng hệ thống nhân sự (Ứng dụng Composition)*

- Tạo lớp Employee(Nhân viên) chứa id, name, salary.
- Tạo lớp Department(Phòng ban) có danh sách Employee.
- Viết chương trình thêm/xóa nhân viên, tính tổng lương phòng ban.

✔️ *Bài tập 4: Sử dụng static và quản lý dữ liệu chung*

Tạo lớp Companychứa biến **static** quản lý tổng số nhân viên.

Viết chương trình tạo nhân viên mới và cập nhật số lượng nhân viên.

📌 **Thử thách**: Viết chương trình quản lý nhân sự nâng cao có menu nhập liệu từ người

dùng.

3 Tổng kết & Giao bài tập về nhà (15 phút)

- **Tóm tắt lại kiến thức chính** về lập trình hướng đối tượng đã thực hành.
- **Nhận xét bài làm của sinh viên, sửa lỗi thường gặp**.
- **Giao bài tập về nhà**: Viết chương trình **quản lý thư viện sách** theo mô hình OOP.

⏳ **Dự kiến phân bổ thời gian:**

🔸 Phần 1 (Ôn tập) – 30 phút

🔸 Phần 2 (Thực hành có hướng dẫn) – 45 phút

🔸 Phần 3 (Bài tập nâng cao) – 45 phút

🔸 Tổng kết & Giao bài tập – 15 phút

📌 **Yêu cầu:** Sinh viên cần **viết mã nguồn, chạy thử, sửa lỗi** và báo cáo kết quả trong buổi học.

PHẦN 2: HƯỚNG DẪN THỰC HÀNH.

Bài tập 1: Quản lý Sản phẩm

Mục tiêu: Sinh viên thực hành tạo lớp đơn giản, sử dụng constructor, getter/setter và anh sách đối tượng.

Bước 1: Xây dựng lớp Product

Tạo lớp Productvới các thuộc tính:

- id (Mã sản phẩm)
- name(Tên sản phẩm)

- price(Giá sản phẩm)
- quantity(Số lượng sản phẩm)

Viết **constructor** để khởi tạo đối tượng.

Cài đặt **getter và setter** để bảo vệ dữ liệu.

📌 *Gợi ý mã nguồn:*

```
public class Product {
    private int id; private String
    name; private double price;
    private int quantity;

    public Product(int id, String name, double price, int quantity) {
        this.id = id;
        this.name = name;
        this.price = price;
        this.quantity = quantity;}

    public int getId() { return id; }
    public String getName() { return name; }
    public double getPrice() { return price; }
    public int getQuantity() { return quantity; }

    public void setPrice(double price) { this.price = price; }
    public void setQuantity(int quantity) {
        this.quantity =quantity; }

    public void displayInfo() {
        System.out.println("ID: " + id + ", Name: " + name + ", Price: " +
            price + ", Quantity: " + quantity);}
}
```

Bước 2: Quản lý danh sách sản phẩm

Tạo lớp ProductManagerđể quản lý danh sách sản phẩm.

Dùng **ArrayList<Product>** để lưu danh sách sản phẩm. Viết các phương thức:

- addProduct(Product p): Thêm sản phẩm mới.
- removeProduct(int id): Xóa sản phẩm theo ID.
- findProduct(int id): Tìm sản phẩm theo ID.

- displayAllProducts(): Hiển thị danh sách sản phẩm.

📌 *Gợi ý mã nguồn:*

```
import java.util.ArrayList;
public class ProductManager {
    private ArrayList<Product> products = new ArrayList<>();

    public void addProduct(Product p) {
        products.add(p);}

    public void removeProduct(int id) {
        products.removeIf(p -> p.getId() == id);}

    public Product findProduct(int id) {
        for (Product p : products) {
            if (p.getId() == id) return p;}
        return null;}
    public void displayAllProducts() {
      for (Product p : products) {
            p.displayInfo();}}
}
```

Bước 3: Chương trình chính để kiểm tra

Tạo lớp Mainvà viết phương thức main().

Tạo đối tượng ProductManager, thêm sản phẩm, hiển thị danh sách.

Thử nghiệm tìm kiếm và xóa sản phẩm.

📌 *Gợi ý mã nguồn:*

```
public class Main {
    public static void main(String[] args) {
        ProductManager manager = new ProductManager();
        manager.addProduct(new Product(1, "Laptop", 1500, 10));
        manager.addProduct(new Product(2, "Mouse", 20, 50));
        System.out.println("Danh sách sản phẩm:");
        manager.displayAllProducts();
        System.out.println("\nTìm sản phẩm có ID 1:");
```

```
        Product p = manager.findProduct(1);
        if (p != null) p.displayInfo();
        System.out.println("\nXóa sản phẩm có ID 2:");
        manager.removeProduct(2);
        manager.displayAllProducts();}
}
```

🔷 **Sinh viên cần chạy thử và kiểm tra kết quả đầu ra.**

Bài tập 2: Kế thừa & Đa hình

Mục tiêu: Sinh viên thực hành kế thừa (extends), ghi đè phương thức (overriding).

Bước 1: Xây dựng lớp cha Vehicle

Tạo lớp Vehiclevới các thuộc tính:

- brand(hãng xe)
- model(mẫu xe)
- year(năm sản xuất)

Viết constructor để khởi tạo đối tượng.

Viết phương thức displayInfo()để in thông tin xe.

📌 *Gợi ý mã nguồn:*

```
public class Vehicle {
    protected String brand;
    protected String model;
    protected int year;

    public Vehicle(String brand, String model, int year) {
        this.brand = brand;
        this.model = model;
        this.year = year;}
    public void displayInfo() {
        System.out.println("Brand: " + brand + ", Model: " + model + ", Year: "
            + year);}
}
```

Bước 2: Xây dựng các lớp con Carvà Motorbike

Lớp Car

- Kế thừa từ Vehicle.
- Thêm thuộc tính numberOfDoors.
- Ghi đè (override) phương thức displayInfo().

📌 Gợi ý mã nguồn:

```
public class Car extends Vehicle {
    private int numberOfDoors;
    public Car(String brand, String model, int year, int numberOfDoors) {
        super(brand, model, year);
        this.numberOfDoors = numberOfDoors;}

    @Override public void displayInfo() {
        super.displayInfo(); System.out.println("Number of Doors: " +
            numberOfDoors);}
}
```

Lớp Motorbike

Kế thừa từ Vehicle.

Thêm thuộc tính hasSidecar.

Ghi đè phương thức displayInfo().

📌 Gợi ý mã nguồn:

```
public class Motorbike extends Vehicle {
    private boolean hasSidecar;
    public Motorbike(String brand, String model, int year, boolean hasSidecar) {
        super(brand, model, year);
        this.hasSidecar = hasSidecar;}

    @Override public void displayInfo() {
        super.displayInfo();
        System.out.println("Has Sidecar: " + hasSidecar);}
}
```

Bước 3: Chương trình chính để kiểm tra

- Tạo lớp Mainvới phương thức main().
- Tạo đối tượng Carvà Motorbike.
- Gọi phương thức displayInfo()để kiểm tra tính kế thừa.

📌 *Gợi ý mã nguồn:*

```
public class Main {
    public static void main(String[] args) {
        Vehicle car = new Car("Toyota", "Camry", 2022, 4);
        Vehicle motorbike = new Motorbike("Honda", "CBR600", 2021, false);

        System.out.println("Thông tin xe hơi:");
        car.displayInfo();
        System.out.println("\nThông tin xe máy:");
        motorbike.displayInfo();}
}
```

PHẦN 3: BÀI TẬP NÂNG CAO.

Bài tập 3: Hệ thống Nhân sự (Composition - HAS-A)

Mục tiêu: Sinh viên thực hành mối quan hệ **Composition (HAS-A)**, sử dụng danh sách đối tượng (ArrayList).

Bước 1: Xây dựng lớp Employee(Nhân viên)

Tạo lớp Employeevới các thuộc tính:

- id (Mã nhân viên)
- name(Tên nhân viên)
- salary(Lương)

Viết constructor để khởi tạo nhân viên.

- Cài đặt **getter và setter** để bảo vệ dữ liệu.
- Viết phương thức displayInfo()để in thông tin nhân viên.

📌 *Gợi ý mã nguồn:*

```
public class Employee {
    private int id;
    private String name;
    private double salary;
```

```
    public Employee(int id, String name, double salary) {
        this.id = id;
        this.name = name;
        this.salary = salary;}

    public int getId() { return id; }
    public String getName() { return name; }
    public double getSalary() { return salary; }

    public void setSalary(double salary) { this.salary = salary; }
    public void displayInfo() {
        System.out.println("ID: " + id + ", Name: " + name + ", Salary: " +
            salary);}
}
```

Bước 2: Xây dựng lớp Department(Phòng ban)

Tạo lớp Departmentvới các thuộc tính:

- name(Tên phòng ban)
- employees(Danh sách nhân viên, sử dụng ArrayList<Employee>)

Viết constructor để khởi tạo phòng ban.

Viết các phương thức:

- addEmployee(Employee e): Thêm nhân viên vào phòng ban.
- removeEmployee(int id): Xóa nhân viên theo ID.
- calculateTotalSalary(): Tính tổng lương của tất cả nhân viên.
- displayAllEmployees(): Hiển thị danh sách nhân viên.

📌 *Gợi ý mã nguồn:*

```
import java.util.ArrayList;
public class Department {
    private String name;
    private ArrayList<Employee> employees;

    public Department(String name) {
        this.name = name;
```

```
        this.employees = new ArrayList<>();}

    public void addEmployee(Employee e) { employees.add(e);}

    public void removeEmployee(int id) {
        employees.removeIf(e -> e.getId() == id);}

    public double calculateTotalSalary() {
        double total = 0;
        for (Employee e : employees) {
            total += e.getSalary();}
        return total;}

    public void displayAllEmployees() {
        System.out.println("Department: " + name);
        for (Employee e : employees) {
            e.displayInfo();}}
}
```

Bước 3: Chương trình chính để kiểm tra

- Tạo lớp Mainvới phương thức main().
- Tạo đối tượng Departmentvà thêm nhân viên vào.
- Hiển thị danh sách nhân viên. Tính tổng lương phòng ban.

Gợi ý mã nguồn:

```
public class Main {
    public static void main(String[] args) {
        Department itDepartment = new Department("IT");
        Employee emp1 = new Employee(1, "Alice", 5000);
        Employee emp2 = new Employee(2, "Bob", 4500);

        itDepartment.addEmployee(emp1);
        itDepartment.addEmployee(emp2);

        System.out.println("Danh sách nhân viên:");
        itDepartment.displayAllEmployees();
```

```
        System.out.println("\nTổng lương phòng IT: " + itDepartment.calculateTotalSalary());
        System.out.println("\nXóa nhân viên có ID 2:");
        itDepartment.removeEmployee(2);
        itDepartment.displayAllEmployees();}
}
```

🔷 **Sinh viên chạy thử và kiểm tra đầu ra của chương trình.**

Bài tập 4: Static và Quản lý Dữ liệu Chung

Mục tiêu: Sinh viên thực hành sử dụng biến và phương thức **static** để quản lý dữ liệu chung.

Bước 1: Xây dựng lớp Company

- Tạo lớp Company với thuộc tính **static**:
  - totalEmployees(Tổng số nhân viên của công ty).
- Viết phương thức **static**:
  - increaseEmployeeCount(): Tăng số lượng nhân viên.
  - decreaseEmployeeCount(): Giảm số lượng nhân viên.
  - getTotalEmployees(): Lấy tổng số nhân viên.

📌 *Gợi ý mã nguồn:*

```
public class Company {
    private static int totalEmployees = 0;

    public static void increaseEmployeeCount() {
        totalEmployees++;}

    public static void decreaseEmployeeCount() {
        totalEmployees--;}

    public static int getTotalEmployees() {
        return totalEmployees;}
}
```

Bước 2: Cập nhật lớp Employeeđể quản lý số lượng nhân viên

Trong constructor của Employee, gọi Company.increaseEmployeeCount(). Khi nhân viên bị xóa, gọi Company.decreaseEmployeeCount().

📌 *Cập nhật mã nguồn trong Employee*

```
public Employee(int id, String name, double salary) {
    this.id = id;
    this.name = name;
    this.salary = salary;
    Company.increaseEmployeeCount();
}
```

📌 *Cập nhật mã nguồn trong Department(khi xóa nhân viên)*

```
public void removeEmployee(int id) {
    employees.removeIf(e -> {
        if (e.getId() == id) {
            Company.decreaseEmployeeCount();
            return true;}
        return false;});
}
```

Bước 3: Chương trình chính để kiểm tra

- Tạo lớp Mainvới phương thức main().
- Tạo nhân viên mới và kiểm tra tổng số nhân viên của công ty.
- Xóa nhân viên và kiểm tra lại tổng số nhân viên.

📌 *Gợi ý mã nguồn:*

```
public class Main {
    public static void main(String[] args) {
        System.out.println("Tổng số nhân viên ban đầu: " +
                Company.getTotalEmployees());

        Employee emp1 = new Employee(1, "Alice", 5000);
        Employee emp2 = new Employee(2, "Bob", 4500);

         System.out.println("Tổng số nhân viên sau khi thêm: " +
                Company.getTotalEmployees());

        Department itDepartment = new Department("IT");
        itDepartment.addEmployee(emp1);
```

```
        itDepartment.addEmployee(emp2);

        System.out.println("\nXóa nhân viên có ID 2:");
        itDepartment.removeEmployee(2);

        System.out.println("Tổng số nhân viên sau khi xóa: " +
            Company.getTotalEmployees());}
    }
```

🔷 **Sinh viên chạy thử và quan sát tổng số nhân viên thay đổi khi thêm hoặc xóa nhân viên.**

BÀI TẬP TRẮC NGHIỆM

CÂU HỎI CƠ BẢN (1 - 10)

Câu 1: Trong Java, từ khóa nào được sử dụng để tạo một lớp?

A. class

B. object

C. method

D. interface

Câu 2: Đối tượng (object) trong Java được tạo ra bằng từ khóa nào?

A. create

B. new

C. object

D. instance

Câu 3: Trong Java, lớp có thể chứa những loại thành phần nào?

A. Biến (fields)

B. Phương thức (methods)

C. Constructor

D. Tất cả các phương án trên

Câu 4: Constructor trong Java có đặc điểm nào?

A. Có cùng tên với lớp

B. Có kiểu trả về void

C. Có thể bị ghi đè (override)

D. Không thể có tham số

Câu 5: Khi nào constructor mặc định được tạo ra bởi Java?

A. Khi lập trình viên không định nghĩa constructor nào.

B. Khi có ít nhất một constructor tham số.

C. Khi có nhiều phương thức trong lớp.

D. Khi lớp không có thuộc tính.

Câu 6: Từ khóa nào trong Java được sử dụng để tham chiếu đến đối tượng hiện tại của lớp?

A. self

B. this

C. current

D. super

Câu 7: Đâu là đặc điểm của phương thức toString() trong Java?

A. Là một phương thức tĩnh (static)

B. Được tự động gọi khi in đối tượng

C. Chỉ có thể sử dụng trong lớp String

D. Không thể ghi đè (override)

Câu 8: Phạm vi truy cập nào giúp bảo vệ dữ liệu trong một lớp nhưng vẫn cho phép truy cập từ lớp con?

A. private

B. public

C. protected

D. default

Câu 9: Điều gì xảy ra nếu không có constructor nào được định nghĩa trong một lớp

A. Java sẽ báo lỗi biên dịch

B. Java sẽ tự động tạo một constructor mặc định

C. Java sẽ không thể tạo đối tượng từ lớp đó

D. Không có gì xảy ra, chương trình vẫn chạy bình thường

Câu 10: Đâu là cú pháp khai báo một đối tượng hợp lệ trong Java?

A. Student s1 = Student();

B. Student s1 = new Student();

C. Student s1();

D. Student s1 = create Student();

CÂU HỎI NÂNG CAO (11 - 20)

Câu 11: Điều gì xảy ra nếu gọi constructor trong một constructor khác của cùng lớp?

E. Lỗi biên dịch

F. Được phép nếu dùng this()

G. Chương trình sẽ bị lỗi runtime

H. Chỉ có thể gọi nếu constructor được khai báo static

Câu 12: Phương thức nào dưới đây có thể được ghi đè (override)?

A. static void display()

B. private void show()

C. final void print()

D. public void getInfo()

Câu 13: Điều gì xảy ra khi gọi super()trong constructor?

A. Gọi constructor của lớp cha

B. Gọi constructor mặc định của lớp hiện tại

C. Gây lỗi biên dịch nếu không có constructor cha

D. Không làm gì cả

Câu 14: Từ khóa finalcó thể áp dụng cho những thành phần nào?

A. Lớp

B. Phương thức

C. Biến

D. Tất cả các phương án trên

Câu 15: Điều gì sẽ xảy ra nếu một lớp kế thừa từ một lớp trừu tượng (abstract class) nhưng không triển khai tất cả các phương thức trừu tượng?

A. Chương trình sẽ biên dịch bình thường

B. Lớp con sẽ phải được khai báo là abstract

C. Lớp con vẫn có thể tạo đối tượng như bình thường

D. Java sẽ tự động triển khai phương thức trừu tượng

Câu 16: Khi nào nên sử dụng từ khóa staticcho một biến trong lớp?

A. Khi biến đó cần được chia sẻ giữa tất cả các đối tượng của lớp

B. Khi biến đó chỉ được sử dụng trong phương thức main()

C. Khi biến đó không cần khởi tạo giá trị

D. Khi biến đó không thể thay đổi giá trị sau khi gán

Câu 17: Điều gì xảy ra khi khai báo một phương thức là statictrong Java?

A. Phương thức có thể truy cập đến biến staticnhưng không thể truy cập đến biến instance

B. Phương thức có thể ghi đè (override) phương thức statickhác

C. Phương thức có thể gọi trực tiếp mà không cần tạo đối tượng của lớp

D. Cả A và C đều đúng

Câu 18: Interface trong Java có đặc điểm nào dưới đây?

A. Không thể chứa phương thức có phần thân (body)

B. Một lớp có thể triển khai (implement) nhiều interface

C. Interface có thể khởi tạo đối tượng trực tiếp

D. Interface có thể chứa thuộc tính private

Câu 19: Điều nào sau đây là đúng về Overloading và Overriding?

A. Overloading xảy ra trong cùng một lớp, còn Overriding xảy ra giữa lớp cha và lớp con

B. Overriding yêu cầu các phương thức có cùng tên nhưng khác số lượng tham số

C. Overloading yêu cầu phương thức con có cùng kiểu trả về với phương thức cha

D. Cả A và C đúng

Câu 20: Giả sử có đoạn mã sau, đầu ra sẽ là gì?

```
class Parent {
    void show() {
        System.out.println("Parent class");}}

class Child extends Parent {
    void show() {
        System.out.println("Child class");}}

public class Test {
    public static void main(String[] args) {
    Parent obj = new Child(); obj.show();}
}
```

A. Parent class

B. Child class

C. Lỗi biên dịch

D. Lỗi runtime

BÀI TẬP TỰ LUYỆN TẬP

Bài tập 1: Tạo lớp Studentđể quản lý sinh viên

Mục tiêu: Ôn tập về khai báo lớp, đối tượng, constructor, getter/setter.

Đề bài

Viết lớp Studentcó các thuộc tính sau:

- id(mã sinh viên - kiểu String)
- name(tên sinh viên - kiểu String)

- gpa(điểm trung bình - kiểu double) Các yêu cầu:

Tạo constructor để khởi tạo giá trị cho sinh viên.

Viết **getter và setter** để truy cập và cập nhật thông tin sinh viên.

Viết phương thức displayInfo()để hiển thị thông tin sinh viên. Trong main(), tạo danh sách 3 sinh viên và hiển thị thông tin của họ.

📌 **Hướng dẫn ngắn gọn**:

- Dùng **private** để bảo vệ dữ liệu.
- Viết **getter/setter** để lấy và cập nhật thông tin.
- Dùng ArrayList<Student>để quản lý danh sách sinh viên trong main().

Bài tập 2: Quản lý tài khoản ngân hàng (BankAccount)

Mục tiêu: Luyện tập **đóng gói dữ liệu (Encapsulation), phương thức nạp/rút tiền**.

Đề bài

Viết lớp BankAccountvới các thuộc tính:

- accountNumber(số tài khoản - kiểu String)
- balance(số dư - kiểu double)

Các yêu cầu:

- Constructor để khởi tạo số tài khoản và số dư.
- Viết **getter** cho số dư (getBalance()).
- Viết phương thức deposit(double amount)để nạp tiền (chỉ cho phép nạp số dương).
- Viết phương thức withdraw(double amount)để rút tiền (chỉ cho phép rút nếu số dư đủ).
- Viết main()để kiểm thử: Tạo tài khoản.
- Nạp và rút tiền.

Hiển thị số dư sau mỗi thao tác.

📌 **Hướng dẫn ngắn gọn**:

- Dùng **private** để bảo vệ số dư.
- Dùng **if** để kiểm tra điều kiện hợp lệ khi nạp/rút tiền.

Bài tập 3: Hệ thống quản lý sản phẩm (Product)

Mục tiêu: Luyện tập **quản lý danh sách đối tượng bằng ArrayList, constructor, phương thức hiển thị thông tin**.

Đề bài

Viết lớp Productvới các thuộc tính:

- id(mã sản phẩm - int)
- name(tên sản phẩm - String)
- price(giá sản phẩm - double)

Các yêu cầu:

- Tạo constructor khởi tạo sản phẩm.
- Viết phương thức displayInfo()để hiển thị thông tin sản phẩm.
- Viết lớp ProductManagerđểquản lý danh sách sản phẩm, với các phương thức:
- addProduct(Product p): Thêm sản phẩm.
- removeProduct(int id): Xóa sản phẩm theo ID.
- displayAllProducts(): Hiển thị danh sách sản phẩm.
- Trong main(), tạo danh sách sản phẩm, thực hiện thêm, xóa và hiển thị danh sách.

📌 **Hướng dẫn ngắn gọn**:

- Dùng ArrayList<Product>để lưu danh sách sản phẩm.
- Kiểm tra idtrước khi xóa để tránh lỗi.

Bài tập 4: Tính diện tích và chu vi các hình học

Mục tiêu: Luyện tập **lập trình hướng đối tượng với kế thừa (Inheritance)**.

Đề bài

Tạo các lớp kế thừa để tính diện tích và chu vi của các hình học:

Lớp cha Shape

Có phương thức calculateArea()và calculatePerimeter()(abstract).

Lớp Rectangle(kế thừa Shape)

Thuộc tính: width, height.

Triển khai phương thức tính diện tích và chu vi.

Lớp Circle(kế thừa Shape)

Thuộc tính: radius.

Triển khai phương thức tính diện tích và chu vi.

Trong main(), tạo đối tượng Rectanglevà Circle, sau đó hiển thị diện tích và chu vi của từng hình.

📌 **Hướng dẫn ngắn gọn:**

Dùng **abstract class** cho lớp Shape.

Override phương thức calculateArea()và calculatePerimeter()trong lớp con.

Bài tập 5: Hệ thống đăng ký khóa học (Enrollment System)

Mục tiêu: Luyện tập **quan hệ giữa các lớp (HAS-A- Composition)**.

Đề bài

Viết chương trình mô phỏng hệ thống đăng ký khóa học, gồm các lớp:

Lớp Student

Thuộc tính: id, name.

Phương thức: displayInfo().

Lớp Course

Thuộc tính: courseId, courseName.
Phương thức: displayCourseInfo().

Lớp Enrollment

Thuộc tính: student, course.

Phương thức displayEnrollment()để hiển thị thông tin đăng ký.

Trong main(), tạo danh sách sinh viên, khóa học, sau đó đăng ký sinh viên vào khóa học và hiển thị danh sách đăng ký.

📌 **Hướng dẫn ngắn gọn**:

Dùng **class Enrollment** để kết nối Studentvà Course.

Khi tạo đối tượng Enrollment, truyền Studentvà Coursevào constructor.

BÀI TẬP LUYỆN TẬP

🔷 Bài tập 1: Tạo lớp Personđể quản lý thông tin cá nhân

Mục tiêu: Ôn tập cách tạo lớp, khai báo thuộc tính và phương thức trong Java.

Đề bài

Viết lớp Personvới các thuộc tính:

- name(Họ và tên, kiểu String)
- age (Tuổi, kiểu int)
- address(Địa chỉ, kiểu String) Các yêu cầu:

Tạo constructor để khởi tạo thông tin.

Viết phương thức displayInfo()để hiển thị thông tin cá nhân. Trong main(), tạo danh sách 3 người và hiển thị thông tin của họ.

📌 **Hướng dẫn**:

Dùng constructor để khởi tạo nhanh dữ liệu.

Dùng ArrayList<Person>để quản lý danh sách.

🔷 Bài tập 2: Quản lý nhân viên (Employee)

Mục tiêu: Thực hành **đóng gói dữ liệu (Encapsulation) và phương thức getter/setter**.

Đề bài

Viết lớp Employeevới các thuộc tính:

- id(Mã nhân viên, kiểu int)
- name(Tên nhân viên, kiểu String)
- salary (Lương, kiểu double)

Các yêu cầu:

- Định nghĩa các **getter và setter** để bảo vệ dữ liệu.
- Viết phương thức increaseSalary(double amount)để tăng lương.
- Trong main(), tạo danh sách nhân viên, cập nhật lương và hiển thị thông tin.

📌 **Hướng dẫn**:

Dùng **private** cho biến instance.

Kiểm tra giá trị hợp lệ trước khi tăng lương.

🔷 Bài tập 3: Mô phỏng ATM (BankAccount)

Mục tiêu: Thực hành **phạm vi truy cập và kiểm soát dữ liệu**.

Đề bài

Viết lớp BankAccountvới các thuộc tính:

- accountNumber(Số tài khoản, kiểu String)
- balance (Số dư, kiểu double)

Các yêu cầu:

- Viết **getter** cho balance.
- Viết phương thức deposit(double amount)để nạp tiền.
- Viết phương thức withdraw(double amount)để rút tiền (chỉ rút khi đủ số dư).
- Trong main(), tạo tài khoản, thực hiện nạp/rút tiền và kiểm tra số dư.

📌 **Hướng dẫn**:

- Dùng **if** để kiểm tra điều kiện hợp lệ trước khi rút tiền.
- In thông báo nếu số dư không đủ.

🔷 Bài tập 4: Quản lý khóa học (Course)

Mục tiêu: Luyện tập **quản lý danh sách đối tượng bằng ArrayList**.

Đề bài

Viết lớp Coursevới các thuộc tính:

- courseId(Mã khóa học, kiểu String)
- courseName(Tên khóa học, kiểu String)
- credit(Số tín chỉ, kiểu int)

Các yêu cầu:

- Tạo danh sách khóa học (ArrayList<Course>).
- Viết phương thức addCourse(Course c)để thêm khóa học.
- Viết phương thức displayCourses()để hiển thị danh sách khóa học.
- Trong main(), tạo danh sách khóa học và hiển thị thông tin.

📌 **Hướng dẫn**:

- Dùng ArrayList<Course>để lưu danh sách.
- Kiểm tra trùng courseIdtrước khi thêm.

🔷 Bài tập 5: Xây dựng lớp Rectangle

Mục tiêu: Thực hành **constructor, getter/setter và phương thức tính toán**.

Đề bài

Viết lớp Rectanglevới các thuộc tính:

- width(Chiều rộng, kiểu double)
- height(Chiều cao, kiểu double) Các yêu cầu:

Viết **constructor** để khởi tạo hình chữ nhật.

Viết phương thức calculateArea()để tính diện tích.

Viết phương thức calculatePerimeter()để tính chu vi.

Trong main(), tạo đối tượng Rectanglevà hiển thị diện tích, chu vi.

📌 **Hướng dẫn:**

Sử dụng công thức:

Diện tích = width * height

Chu vi = (width + height) * 2

🔷 Bài tập 6: Mô phỏng quản lý thư viện (Library)

Mục tiêu: Luyện tập **tương tác giữa các lớp (Composition - HAS-A)**.

📌 **Gợi ý**:

- Dùng **class Book**để đại diện sách.
- Dùng **class Library**để quản lý danh sách sách.

- Viết phương thức thêm/xóa/tìm kiếm sách.

🔷 Bài tập 7: Tạo lớp Carvà Motorbike(Kế thừa - IS- A)

Mục tiêu: Thực hành kế thừa (Inheritance) và ghi đè phương thức (Override).

📌 **Gợi ý**:

Dùng **lớp cha Vehicle**để chứa thông tin chung (brand, model, year).

Tạo lớp con **Car**và **Motorbike**, ghi đè phương thức displayInfo()

🔷 Bài tập 8: Sử dụng từ khóa staticđể đếm số lượng đối tượng

Mục tiêu: Hiểu cách dùng **static trong Java**.

📌 **Gợi ý**:

Dùng **biến static count**để đếm số lượng đối tượng được tạo.
Viết chương trình kiểm tra số lượng đối tượng sau khi khởi tạo.

🔷 Bài tập 9: Tạo lớp Shapevà kế thừa Rectangle, Circle

Mục tiêu: Luyện tập đa hình (Polymorphism) và lớp trừu tượng (Abstract class).

📌 **Gợi ý**:

Dùng **abstract class Shape**để chứa phương thức calculateArea().

Kế thừa lớp Shapevà triển khai phương thức tính diện tích trong Rectanglevà Circle.

🔷 Bài tập 10: Viết chương trình quản lý sinh viên (StudentManager)

Mục tiêu: Luyện tập quản lý danh sách đối tượng và tìm kiếm dữ liệu.

📌 **Gợi ý**:

Dùng **ArrayList<Student>** để quản lý danh sách.

Viết phương thức searchStudent(int id)để tìm sinh viên theo ID.

BÀI TẬP DỰ ÁN (BÀI TẬP NHÓM)

💡 **Mục tiêu chung:**

✓ Phát triển kỹ năng lập trình hướng đối tượng (OOP) theo dự án thực tế.

✓ Rèn luyện làm việc nhóm, phân công công việc và viết mã nguồn theo chuẩn.

✓ Tạo sản phẩm hoàn chỉnh có thể mở rộng và cải tiến.

🌟 **Tiêu chí đánh giá chung cho các dự án**

📌 **Chất lượng mã nguồn** (40%)

Tuân thủ nguyên tắc OOP: **Encapsulation, Inheritance, Polymorphism, Composition**.
Code dễ đọc, có ghi chú (comment).

📌 **Hoạt động của chương trình** (30%)

Chạy đúng chức năng theo yêu cầu.

Kiểm tra lỗi đầu vào và xử lý ngoại lệ hợp lý.

📌 **Báo cáo & Thuyết trình** (30%)

Trình bày **sơ đồ UML** hoặc mô tả quan hệ giữa các lớp.
Hướng dẫn cách sử dụng chương trình.

Gợi ý cải tiến hệ thống trong tương lai.

🔷 Dự án 1: Hệ thống Quản lý Sinh viên (Student Management System)

Mục tiêu: Áp dụng lớp và đối tượng, danh sách đối tượng (ArrayList), đọc/ghi file.

📌 Yêu cầu

Phát triển hệ thống quản lý sinh viên với các chức năng chính:

Quản lý sinh viên:

- Thêm sinh viên.
- Xóa sinh viên theo ID.
- Cập nhật thông tin sinh viên.
- Tìm kiếm sinh viên theo tên hoặc ID.

Quản lý khóa học:

- Thêm/xóa khóa học.
- Đăng ký sinh viên vào khóa học.
- Xem danh sách sinh viên của một khóa học.

Lưu trữ dữ liệu:

- Lưu danh sách sinh viên vào file (students.txt).
- Đọc dữ liệu từ file khi khởi động chương trình.

📌 Hướng dẫn thực hiện

🏷️ *Giai đoạn 1: Thiết kế hệ thống*

Xác định các lớp chính:

Student(Mã SV, Tên, Ngành, Điểm GPA).

Course(Mã khóa học, Tên, Số tín chỉ).
StudentManagerđể quản lý danh sách sinh viên. Vẽ
sơ đồ quan hệ giữa các lớp (HAS-A, IS-A).

Giai đoạn 2: Lập trình

Cài đặt các lớp và phương thức.

Dùng ArrayList<Student>để quản lý sinh viên.

Dùng BufferedReadervà BufferedWriterđể đọc/ghi file.

Giai đoạn 3: Hoàn thiện & Kiểm thử *Viết giao diện dòng lệnh cho người dùng (Scanner).*

Chạy thử nghiệm với nhiều trường hợp khác nhau.

Dự án 2: Ứng dụng Quản lý Nhà hàng (Restaurant Management System)

Mục tiêu: Áp dụng lập trình hướng đối tượng, kế thừa (Inheritance), đa hình (Polymorphism).

Yêu cầu

Xây dựng hệ thống quản lý nhà hàng với các chức năng:

Quản lý món ăn:

Thêm/xóa món ăn.

Hiển thị danh sách món ăn theo danh mục.

Quản lý đơn hàng:

Thêm món ăn vào đơn hàng.

Hiển thị hóa đơn và tính tổng tiền.

Nhân viên và khách hàng:

- Employee: Quản lý thông tin nhân viên.
- Customer: Lưu thông tin khách hàng thân thiết.

Hướng dẫn thực hiện

Giai đoạn 1: Thiết kế hệ thống

Xác định các lớp chính:

- Food(Tên, Loại, Giá).
- Order(Danh sách món ăn, Tổng tiền).
- Employee(Nhân viên, Lương).
- Customer(Tên, Số điện thoại, Điểm tích lũy).

Áp dụng **kế thừa** (Managervà Waiterkế thừa Employee).

Giai đoạn 2: Lập trình

Dùng **ArrayList<Food>** để lưu danh sách món ăn.

Dùng **ArrayList<Order>** để quản lý đơn hàng.

Ghi đè (Override) phương thức calculateSalary()trong Managervà Waiter.

Giai đoạn 3: Kiểm thử & Báo cáo

Kiểm tra hiển thị danh sách món ăn, thêm đơn hàng.

Nhóm trình bày giải pháp và hướng mở rộng.

Dự án 3: Hệ thống Đặt Vé Xe Bus (Bus Ticket Booking System)

Mục tiêu: Áp dụng quan hệ giữa các lớp (HAS-A, Composition), nhập/xuất dữ liệu từ file.

Yêu cầu

Hệ thống đặt vé xe bus gồm các chức năng chính:

Quản lý chuyến xe:

Thêm/xóa chuyến xe. Xem thông tin chuyến xe.

Đặt vé:

Chọn chuyến xe theo mã. Kiểm tra số ghế còn trống. Xác nhận đặt vé.

Lưu trữ dữ liệu:

Lưu danh sách chuyến xe vào file.

Đọc dữ liệu từ file khi khởi động chương trình.

Hướng dẫn thực hiện

Giai đoạn 1: Thiết kế hệ thống

Xác định các lớp chính:

- Bus(Mã xe, Tuyến đường, Số ghế).
- Ticket(Mã vé, Hành khách, Chuyến xe).
- BookingSystemđể quản lý đặt vé.

Sử dụng **Composition (HAS-A)**: BookingSystemchứa ArrayList<Ticket>.

Giai đoạn 2: Lập trình

- Dùng ArrayList<Bus>để quản lý danh sách xe.
- Dùng ArrayList<Ticket>để lưu thông tin vé.
- Dùng BufferedReadervà BufferedWriterđể đọc/ghi file.

Giai đoạn 3: Kiểm thử & Báo cáo

Thử nghiệm đặt vé, kiểm tra ghế trống.

Nhóm trình bày thiết kế và hướng phát triển tiếp theo.