

BUỔI 6: TÍNH ĐÓNG GÓI VÀ CHE GIẤU THÔNG TIN TRONG OOP

Thời lượng: 3 tiết

I. Mục tiêu

Sau buổi học, sinh viên sẽ:

- ✓ Hiểu rõ khái niệm **đóng gói (Encapsulation)** và lý do sử dụng.
- ✓ Phân biệt được **các mức độ truy cập** (private, public, protected, default).
- ✓ Biết cách sử dụng **getter và setter** để bảo vệ dữ liệu trong lớp.
- ✓ Viết được chương trình **ứng dụng tính đóng gói** để quản lý thông tin.

II. Nội dung chi tiết

◆ Giới thiệu về tính đóng gói

1. Mở đầu và Ôn tập

- **Hỏi đáp ôn tập về lớp và đối tượng:**
 - Class và Object là gì?
 - Tại sao cần bảo vệ dữ liệu của một đối tượng?
- **Tình huống thực tế:**

Bạn có muốn ai đó có thể thay đổi số dư tài khoản ngân hàng của bạn mà không có sự kiểm soát không?

→ **Giới thiệu về Encapsulation:** Bảo vệ dữ liệu và chỉ cho phép truy cập thông qua phương thức được kiểm soát.

2. Lý thuyết về Tính đóng gói (Encapsulation)

- **Định nghĩa:**
 - Che giấu dữ liệu bên trong lớp và chỉ cho phép truy cập qua các phương thức công khai.
- **Lợi ích của Encapsulation:**
 - ✓ Bảo vệ dữ liệu tránh sửa đổi trái phép.
 - ✓ Giúp dễ dàng kiểm soát dữ liệu nhập vào.
 - ✓ Hỗ trợ bảo trì và mở rộng chương trình.
- **Mức độ truy cập trong Java:**

Mức truy cập	Trong cùng lớp	Trong cùng package	Lớp con (khác package)	Bất kỳ đâu
private	✓	✗	✗	✗
default	✓	✓	✗	✗

Mức truy cập	Trong cùng lớp	Trong cùng package	Lớp con (khác package)	Bất kỳ đâu
protected	✓	✓	✓	✗
public	✓	✓	✓	✓

3. Ví dụ minh họa

- **Code trước khi sử dụng Encapsulation (Lỗi có thể xảy ra)**

```
class Account {
    String owner;
    double balance;
}
public class Main {
    public static void main(String[] args) {
        Account acc = new Account();
        acc.owner = "Nguyen Van A";
        acc.balance = -500; // LỖI: Không có kiểm tra giá trị âm
        System.out.println(acc.owner + " có số dư: " + acc.balance);
    }
}
```

✗ **Vấn đề:** Người dùng có thể nhập số dư âm mà không bị kiểm soát.

- **Code sau khi sử dụng Encapsulation (Bảo vệ dữ liệu)**

```
class Account {
    private String owner;
    private double balance;
    public Account(String owner, double balance) {
        this.owner = owner;
        if (balance >= 0) {
            this.balance = balance;
        } else {
            this.balance = 0;
        }
    }
    public double getBalance() {
        return balance;
    }
    public void deposit(double amount) {
        if (amount > 0) {
            balance += amount;
        }
    }
    public void withdraw(double amount) {
        if (amount > 0 && amount <= balance) {
            balance -= amount;
        }
    }
}
```

```
    } else {  
        System.out.println("Không thể rút số tiền lớn hơn số dff!");  
    }  
}
```

✅ **Kết quả:** Dữ liệu được bảo vệ, không thể đặt số dư âm.

🔹 Thực hành lập trình tính đóng gói

1. Bài tập thực hành cá nhân

- **Yêu cầu:** Viết chương trình quản lý sinh viên với các thuộc tính:
 - name (Tên sinh viên, kiểu String).
 - id (Mã sinh viên, kiểu String).
 - GPA (Điểm trung bình, kiểu double).
- **Yêu cầu kỹ thuật:**
 - Đóng gói dữ liệu bằng private.
 - Cung cấp phương thức getter và setter để truy xuất dữ liệu an toàn.
 - Kiểm tra tính hợp lệ khi cập nhật GPA (chỉ nhận giá trị từ 0.0 đến 4.0).
- **Code mẫu gợi ý:**

```
class Student {  
    private String name;  
    private String id;  
    private double GPA;  
  
    public Student(String name, String id, double GPA){  
        this.name = name;  
        this.id = id;  
        setGPA(GPA);  
    }  
    public String getName() {  
        return name;  
    }  
    public void setName(String name) {  
        this.name = name;  
    }  
    public String getId() {  
        return id;  
    }  
    public double getGPA() {  
        return GPA;  
    }  
    public void setGPA(double GPA) {
```

```
if (GPA >= 0.0 && GPA <= 4.0) {  
    this.GPA = GPA;  
} else {  
    System.out.println("Điểm GPA không hợp lệ!");}}}
```

2. Chữa bài & Thảo luận chung

- Kiểm tra lỗi phổ biến:
 - Không sử dụng private → Bị thay đổi dữ liệu ngoài mong muốn.
 - Không kiểm tra giá trị trong setter → Nhập điểm GPA không hợp lệ.
- **Hỏi đáp:** Nếu không có getter/setter, hệ thống có thể gặp vấn đề gì?

◆ Bài tập nhóm & Ứng dụng thực tế

1. Bài tập nhóm: Xây dựng chương trình quản lý nhân viên

- **Yêu cầu:** Viết chương trình quản lý nhân viên (Employee).
 - Thuộc tính: name, salary, id (tự động tăng nhờ static).
 - **Quy định:**
 - Dữ liệu được đóng gói bằng private.
 - Cung cấp getter và setter để kiểm soát truy cập.
 - Lớp có biến static để đếm số nhân viên.
- **Gợi ý Code:**

```
class Employee {  
    private static int count = 0;  
    private String name;  
    private double salary;  
    private int id;  
  
    public Employee(String name, double salary) {  
        this.name = name;  
        this.salary = salary;  
        this.id = ++count;}  
    public int getId() {  
        return id;}  
    public String getName() {  
        return name;}  
    public void setName(String name) {  
        this.name = name;}  
}
```

```
public double getSalary() {  
    return salary;}  
public void setSalary(double salary) {  
    if (salary > 0) {  
        this.salary = salary;}}  
public static int getEmployeeCount() {  
    return count;}}
```

2. Tổng kết & Hỏi đáp

- **Chữa bài tập nhóm.**
- **Thảo luận:** Khi nào nên dùng private, protected, public?
- **Nhấn mạnh kiến thức quan trọng:** ☒ Bảo vệ dữ liệu bằng private.
☒ Kiểm soát truy cập bằng getter/setter.
☒ static giúp quản lý dữ liệu dùng chung cho tất cả đối tượng.

Tóm tắt kiến thức trọng tâm

- ☒ **Đóng gói** giúp bảo vệ dữ liệu.
- ☒ **Getter & Setter** kiểm soát truy cập dữ liệu.
- ☒ **Mức độ truy cập** quyết định phạm vi sử dụng biến.
- ☒ Thực hành giúp nắm vững OOP hơn! 🎯

III. BÀI TẬP CÓ LỜI GIẢI

🔗 Bài tập 1: Quản lý danh sách Sản phẩm (Product Management)

Đề bài

Viết một chương trình **quản lý danh sách sản phẩm** trong cửa hàng bằng cách sử dụng **Encapsulation (đóng gói)**.

Yêu cầu

1. Tạo lớp Product có các thuộc tính sau:
 - id (Mã sản phẩm, kiểu int, tự động tăng).
 - name (Tên sản phẩm, kiểu String).
 - price (Giá sản phẩm, kiểu double, không được âm).
2. Viết các phương thức:
 - Constructor để khởi tạo sản phẩm.
 - getter cho tất cả các thuộc tính.

- setter cho name và price, kiểm tra hợp lệ.
3. Viết chương trình chính (main) để:
- Tạo danh sách các sản phẩm.
 - Thêm sản phẩm mới vào danh sách.
 - Hiển thị danh sách sản phẩm ra màn hình.

Hướng dẫn giải

1. Xây dựng lớp Product với Encapsulation

- Đóng gói (private) dữ liệu để tránh thay đổi tùy tiện.
- Sử dụng static để tự động tăng ID.
- Kiểm tra dữ liệu hợp lệ khi cập nhật giá sản phẩm.

```
class Product {  
    private static int count = 0; // Tự động tăng ID  
  
    private int id;  
    private String  
    name; private  
    double price;  
  
    public Product(String name, double price)  
    { this.id = ++count;  
      this.name = name;  
      setPrice(price); } // Kiểm tra giá trị hợp lệ  
  
    public int getId() {  
        return id;}  
    public String getName() {  
        return name;}  
    public void setName(String name) {  
        this.name = name;}  
    public double getPrice() {  
        return price;}  
    public void setPrice(double price) {  
        if (price > 0) {  
            this.price = price;  
        } else {  
            System.out.println("Giá sản phẩm phải lớn hơn 0!");  
        }  
    }  
    public void display() {  
        System.out.println("ID: " + id + ", Name: " + name +  
        ", Price: " + price);  
    }  
}
```

```
}
```

2. Viết chương trình Main

- Tạo danh sách sản phẩm.
- Thêm sản phẩm vào danh sách.
- Hiển thị danh sách sản phẩm.

```
import java.util.ArrayList;
public class Main {
    public static void main(String[] args) {
        ArrayList<Product> products = new
        ArrayList<>();
        // Thêm sản phẩm vào danh sách
        products.add(new Product("Laptop", 1500.0));
        products.add(new Product("Smartphone", 800.0));
        products.add(new Product("Headphone", 100.0));
        // Hiển thị danh sách sản phẩm
        System.out.println("Danh sách sản phẩm:");

        for (Product p : products) {
            p.display();
        }
    }
}
```

Bài tập 2: Hệ thống Quản lý Lớp học (Class Management)

Đề bài

Xây dựng chương trình **quản lý lớp học** với danh sách sinh viên, áp dụng **Encapsulation (đóng gói), Getter/Setter và Static**.

Yêu cầu

1. Tạo lớp Student có các thuộc tính sau:
 - id (Mã sinh viên, tự động tăng).
 - name (Tên sinh viên).
 - grade (Điểm số, từ 0 đến 10).
2. Viết phương thức:
 - Constructor để khởi tạo sinh viên.
 - getter và setter với kiểm tra dữ liệu hợp lệ.
3. Viết chương trình chính (main) để:
 - Tạo danh sách sinh viên.
 - Hiển thị danh sách sinh viên.

Hướng dẫn giải

1. Xây dựng lớp Student

```
class Student {  
    private static int  
    count = 0; private int  
    id;  
    private String  
    name; private  
    double grade;  
  
    public Student(String name, double grade)  
    { this.id = ++count;  
      this.name = name;  
      setGrade(grade); } // Kiểm tra điểm hợp lệ  
  
    public int getId()  
    { return id;}  
    public String getName()  
    { return name;}  
    public void setName(String name) {  
        this.name = name;}  
    public double  
        getGrade() { return  
            grade;}  
    public void setGrade(double  
        grade) { if (grade >= 0 &&  
        grade <= 10) {  
            this.grade = grade;  
        } else {  
            System.out.println("Điểm số phải từ 0 đến 10!");  
        }  
    }  
  
    public void display() {  
        System.out.println("ID: " + id + ", Name: " + name + ", Grade:  
        " + grade);}  
}
```

2. Viết chương trình Main

```
import java.util.ArrayList;
```



```

public class Main {
    public static void main(String[] args) {
        ArrayList<Student> students = new
        ArrayList<>();
        // Thêm sinh viên vào danh sách
        students.add(new Student("Nguyen Van A", 8.5));
        students.add(new Student("Tran Thi B", 7.0));
        students.add(new Student("Le Van C", 9.2));
        // Hiển thị danh sách sinh viên
        System.out.println("Danh sách sinh viên:");

        for (Student s : students) {
            s.display();}}
    }
}

```

Bài tập 3: Quản lý Nhân viên và Tính Lương

Đề bài

Viết chương trình **quản lý nhân viên** của một công ty, trong đó mỗi nhân viên có **lương cơ bản** và **thưởng**, tổng lương được tính dựa trên công thức:

Tổng lương=Lương cơ bản+Thưởng

Yêu cầu

1. Tạo lớp Employee với các thuộc tính:
 - id (Mã nhân viên, tự động tăng).
 - name (Tên nhân viên).
 - baseSalary (Lương cơ bản, không âm).
 - bonus (Thưởng, không âm).
2. Viết phương thức:
 - Constructor khởi tạo nhân viên.
 - getter và setter với kiểm tra dữ liệu hợp lệ.
 - getTotalSalary() để tính tổng lương.
3. Viết chương trình main để:
 - Thêm nhân viên.
 - Hiển thị danh sách nhân viên và tổng lương của họ.

Hướng dẫn giải

1. Xây dựng lớp Employee

```

class Employee {
    private static int count = 0;
    private int id;
    private String name;

```

```

private double
baseSalary;
private double bonus;
public Employee(String name, double baseSalary, double bonus) {
    this.id = ++count;
    this.name = name;
    setBaseSalary(baseSalary);
    setBonus(bonus);}
public double getTotalSalary() {
    return baseSalary + bonus;}
public void setBaseSalary(double baseSalary) {
    if (baseSalary >= 0) {
        this.baseSalary = baseSalary;}}

public void setBonus(double
bonus) { if (bonus >= 0) {
    this.bonus = bonus;}
}

public void display() {
    System.out.println("ID: " + id + ", Name: " +
name + ", Total Salary: " + getTotalSalary());}
}

```


2. Viết chương trình Main

```

public class Main {

    public static void main(String[] args) {
        Employee e1 = new Employee("Alice", 5000, 1000);
        Employee e2 = new Employee("Bob", 6000, 500);
        e1.display();
        e2.display();}
}

```

 **Kết quả:** Hiển thị danh sách nhân viên với tổng lương của họ.

IV. CÂU HỎI TRẮC NGHIỆM

Phần 1: Câu hỏi về Lập trình Hướng Đối Tượng (OOP)

Câu 1

Lập trình hướng đối tượng (OOP) có mấy đặc trưng chính?

- A. 2
- B. 3
- C. 4
- D. 5

Câu 2

Trong OOP, đặc trưng nào giúp che giấu dữ liệu và chỉ cho phép truy cập thông qua các phương thức?

- A. Đa hình (Polymorphism)
- B. Kế thừa (Inheritance)
- C. Đóng gói (Encapsulation)
- D. Trừu tượng (Abstraction)

Câu 3

Lớp (class) là gì trong Java?

- A. Một bản thiết kế cho đối tượng
- B. Một phương thức trong Java
- C. Một kiểu dữ liệu nguyên thủy
- D. Một biến chứa nhiều giá trị

Câu 4

Từ khóa nào được sử dụng để tạo một đối tượng trong Java?

- A. class
- B. static
- C. new
- D. object

Câu 5

Phạm vi truy cập nào giúp một thuộc tính chỉ có thể được truy cập trong cùng một lớp?

- A. public
- B. private
- C. protected
- D. default

Phần 2: Câu hỏi về Lớp và Đối tượng

Câu 6

Đối tượng (Object) trong Java là gì?

- A. Một bản thiết kế để tạo ra lớp
- B. Một thể hiện cụ thể của một lớp
- C. Một phương thức để truy xuất dữ liệu
- D. Một biến toàn cục

Câu 7

Constructor (hàm khởi tạo) có nhiệm vụ gì?

- A. Khởi tạo giá trị cho đối tượng khi được tạo
- B. Hủy một đối tượng trong Java
- C. Gọi phương thức main() trong lớp
- D. Kiểm tra kiểu dữ liệu của biến

Câu 8

Constructor có thể có kiểu dữ liệu trả về không?

- A. Có, bất kỳ kiểu dữ liệu nào
- B. Không, constructor không có kiểu dữ liệu trả về
- C. Chỉ có thể là kiểu void
- D. Chỉ có thể là kiểu int

Câu 9

Lệnh nào sau đây tạo một đối tượng hợp lệ của lớp Student?

- A. Student s1 = Student();
- B. Student s1 = new Student();
- C. Student s1;
- D. new Student s1();

Câu 10

Nếu không khai báo phạm vi truy cập, phạm vi mặc định của thuộc tính trong Java là gì?

- A. private
- B. public
- C. protected
- D. default

Phần 3: Câu hỏi về Đóng gói và Getter/Setter

Câu 11

Tại sao nên sử dụng private cho các thuộc tính của lớp?

- A. Giúp bảo vệ dữ liệu và tránh truy cập trái phép
- B. Giúp phương thức main() dễ truy cập hơn
- C. Giúp chương trình chạy nhanh hơn
- D. Không có tác dụng gì đặc biệt

Câu 12

Phương thức nào sau đây được dùng để lấy giá trị của một thuộc tính private trong Java?

- A. Setter
- B. Getter
- C. Constructor
- D. Static method

Câu 13

Phương thức setter có nhiệm vụ gì?

- A. Dùng để lấy giá trị của một thuộc tính private
- B. Dùng để thiết lập giá trị cho một thuộc tính private
- C. Dùng để xóa giá trị của một thuộc tính
- D. Dùng để tạo đối tượng mới

Câu 14

Điều gì xảy ra nếu một thuộc tính được khai báo private nhưng không có getter hoặc setter?

- A. Có thể truy cập trực tiếp từ bất kỳ lớp nào
- B. Chỉ có thể truy cập trong cùng lớp đó
- C. Có thể truy cập từ lớp con
- D. Có thể truy cập trong cùng package

Câu 15

Giả sử có lớp Student với thuộc tính private int age;. Phương thức nào dưới đây đúng để cập nhật tuổi của sinh viên?

- A. void setAge(int age) { this.age = age; }
- B. public void age(int newAge) { age = newAge; }
- C. setAge(int newAge) { this.age = newAge; }
- D. void setAge() { return age; }

Phần 4: Câu hỏi về Static và This**Câu 16**

Từ khóa this được sử dụng để làm gì trong Java?

- A. Tham chiếu đến lớp cha
- B. Tham chiếu đến đối tượng hiện tại của lớp
- C. Tham chiếu đến phương thức main()
- D. Tham chiếu đến biến static

Câu 17

Lệnh nào sau đây là một cách sử dụng đúng của this trong constructor?

- A. this = new Student();
- B. this.name = name;

C. `this.Student(name, age);`

D. `this -> name = name;`

Câu 18

Từ khóa `static` có thể áp dụng cho thành phần nào sau đây?

A. Biến

B. Phương thức

C. Khối lệnh

D. Cả A, B và C đều đúng

Câu 19

Điều gì xảy ra nếu một biến được khai báo là `static`?

A. Biến sẽ được chia sẻ giữa tất cả các đối tượng của lớp

B. Mỗi đối tượng có một bản sao riêng của biến đó

C. Biến chỉ có thể được truy cập từ phương thức `main()`

D. Biến sẽ bị xóa khi chương trình kết thúc

Câu 20

Làm thế nào để truy cập một biến `static` từ bên ngoài lớp mà không cần tạo đối tượng?

A. `this.variableName;`

B. `ClassName.variableName;`

C. `new ClassName().variableName;`

D. `objectName.variableName;`

V. BÀI TẬP TỰ LÀM

Bài tập 1: Quản lý Thư viện (Library Management)

Viết chương trình **quản lý sách trong thư viện** sử dụng **Lập trình hướng đối tượng (OOP)** và **Đóng gói (Encapsulation)**.

Yêu cầu

- Tạo lớp `Book` với các thuộc tính:
 - `title` (Tiêu đề sách, kiểu `String`).
 - `author` (Tác giả, kiểu `String`).
 - `isBorrowed` (Trạng thái mượn, kiểu `boolean`, mặc định là `false`).
- Viết các phương thức:
 - Constructor để khởi tạo sách.
 - getter cho tất cả các thuộc tính.
 - `borrowBook()` để đánh dấu sách là đã mượn.
 - `returnBook()` để đánh dấu sách là có sẵn.

3. Viết chương trình main để:
 - Tạo danh sách sách.
 - Cho phép mượn và trả sách.
 - Hiển thị danh sách sách trong thư viện.

Bài tập 2: Quản lý Tài khoản Ngân hàng (Bank Account Management)

Viết chương trình **quản lý tài khoản ngân hàng**, áp dụng **Encapsulation, Getter/Setter và Static**.

Yêu cầu

1. Tạo lớp BankAccount với các thuộc tính:
 - accountNumber (Số tài khoản, tự động tăng).
 - ownerName (Tên chủ tài khoản).
 - balance (Số dư tài khoản, không được âm).
2. Viết các phương thức:
 - Constructor khởi tạo tài khoản với số dư ≥ 0 .
 - getter cho tất cả các thuộc tính.
 - deposit(double amount) để nạp tiền.
 - withdraw(double amount) để rút tiền (kiểm tra số dư đủ mới được rút).
 - getAccountCount() để trả về số lượng tài khoản đã tạo.
3. Viết chương trình main để:
 - Tạo danh sách tài khoản.
 - Nạp và rút tiền từ tài khoản.
 - Hiển thị danh sách tài khoản và số dư.

Bài tập 3: Quản lý Học sinh và Điểm số (Student Grade Management)

Viết chương trình **quản lý học sinh và điểm số**, áp dụng **Encapsulation, Getter/Setter, Static và This**.

Yêu cầu

1. Tạo lớp Student với các thuộc tính:
 - id (Mã học sinh, tự động tăng).
 - name (Tên học sinh).
 - mathScore, englishScore, scienceScore (Điểm số môn Toán, Anh, Khoa học, từ 0 - 10).
2. Viết các phương thức:
 - Constructor để khởi tạo học sinh.

- getter và setter cho các thuộc tính (kiểm tra điểm hợp lệ từ 0 - 10).
 - calculateAverage() để tính điểm trung bình.
 - displayInfo() để hiển thị thông tin học sinh.
3. Viết chương trình main để:
- Thêm học sinh vào danh sách.
 - Cập nhật điểm số.
 - Hiển thị danh sách học sinh và điểm trung bình.

VI. BÀI TẬP LUYỆN TẬP

Bài tập 1: Quản lý Đơn hàng (Order Management)

Viết chương trình **quản lý đơn hàng**, áp dụng **Encapsulation, Getter/Setter và Static**.

Yêu cầu

1. Tạo lớp Order với các thuộc tính:
 - orderId (Mã đơn hàng, tự động tăng).
 - customerName (Tên khách hàng).
 - totalAmount (Tổng số tiền, không được âm).
 - status (Trạng thái đơn hàng: "Pending", "Shipped", "Delivered").
2. Viết các phương thức:
 - Constructor để khởi tạo đơn hàng.
 - getter và setter cho các thuộc tính (kiểm tra giá trị hợp lệ).
 - updateStatus(String newStatus) để cập nhật trạng thái đơn hàng (chỉ cho phép 3 trạng thái trên).
3. Viết chương trình main để:
 - Tạo danh sách đơn hàng.
 - Cập nhật trạng thái đơn hàng.
 - Hiển thị danh sách đơn hàng.

Bài tập 2: Quản lý Xe ô tô (Car Management)

Viết chương trình **quản lý xe ô tô**, áp dụng **Encapsulation, Static và This**.

Yêu cầu

1. Tạo lớp Car với các thuộc tính:
 - carId (Mã xe, tự động tăng).
 - brand (Hãng xe: "Toyota", "Honda", "Ford"...).
 - model (Dòng xe, ví dụ: "Camry", "Civic"...).
 - year (Năm sản xuất, phải từ 1990 trở đi).
 - price (Giá xe, không âm).

2. Viết các phương thức:
 - Constructor để khởi tạo xe.
 - getter và setter cho các thuộc tính (kiểm tra giá trị hợp lệ).
 - displayInfo() để hiển thị thông tin xe.
3. Viết chương trình main để:
 - Tạo danh sách xe.
 - Cập nhật thông tin xe.
 - Hiển thị danh sách xe.

Bài tập 3: Quản lý Nhân viên trong Công ty (Company Employee Management)

Viết chương trình **quản lý nhân viên trong công ty**, áp dụng **Encapsulation, Getter/Setter, Static và This**.

Yêu cầu

1. Tạo lớp Employee với các thuộc tính:
 - employeeId (Mã nhân viên, tự động tăng).
 - name (Tên nhân viên).
 - department (Bộ phận làm việc: "IT", "HR", "Finance", "Marketing"...).
 - salary (Mức lương, phải lớn hơn 3 triệu).
2. Viết các phương thức:
 - Constructor để khởi tạo nhân viên.
 - getter và setter cho các thuộc tính (kiểm tra giá trị hợp lệ).
 - displayInfo() để hiển thị thông tin nhân viên.
3. Viết chương trình main để:
 - Tạo danh sách nhân viên.
 - Cập nhật thông tin nhân viên.
 - Hiển thị danh sách nhân viên.

VII. BÀI TẬP DỰ ÁN

Dự án Nhóm: Hệ thống Quản lý Cửa hàng Điện tử (Electronics Store Management System)

Số lượng sinh viên: Nhóm 2 người

Mục tiêu:

- ✓ Thực hành **Encapsulation, Getter/Setter, Static, This**.
- ✓ Làm việc với **danh sách đối tượng** (ArrayList).
- ✓ Sử dụng **phân công công việc nhóm** hiệu quả.

Yêu cầu dự án

Hãy xây dựng một chương trình **quản lý cửa hàng điện tử**, trong đó cửa hàng bán các sản phẩm điện tử như **Laptop, Điện thoại, Máy tính bảng, Tai nghe, TV**.

Chương trình sẽ có các chức năng:

1. **Quản lý Sản phẩm** (Product Management): Thêm, xóa, cập nhật thông tin sản phẩm.
2. **Quản lý Đơn hàng** (Order Management): Tạo đơn hàng, hiển thị danh sách đơn hàng.
3. **Báo cáo Doanh thu**: Tính tổng doanh thu từ các đơn hàng.

Phân công công việc nhóm

Người 1: Xây dựng lớp Product và các chức năng liên quan đến sản phẩm.

Người 2: Xây dựng lớp Order và các chức năng liên quan đến đơn hàng.

Cả hai: Hợp tác để viết lớp Main, tích hợp hệ thống và kiểm thử.

Chi tiết triển khai

1. Lớp Product (Quản lý sản phẩm)

- **Thuộc tính:**
 - productId (Mã sản phẩm, tự động tăng).
 - name (Tên sản phẩm).
 - category (Loại sản phẩm: "Laptop", "Phone", "Tablet", "Headphone", "TV").
 - price (Giá sản phẩm, phải lớn hơn 0).
 - stock (Số lượng tồn kho, không âm).
- **Phương thức:**
 - Constructor để khởi tạo sản phẩm.
 - getter và setter có kiểm tra hợp lệ.
 - displayInfo() để hiển thị thông tin sản phẩm.

2. Lớp Order (Quản lý đơn hàng)

- **Thuộc tính:**
 - orderId (Mã đơn hàng, tự động tăng).
 - customerName (Tên khách hàng).
 - orderedProducts (Danh sách sản phẩm đã đặt hàng).
 - totalPrice (Tổng tiền đơn hàng, tự động tính).
- **Phương thức:**
 - Constructor để tạo đơn hàng.
 - getter và setter.
 - calculateTotalPrice() để tính tổng tiền đơn hàng.

- `displayOrder()` để hiển thị thông tin đơn hàng.

3. Chức năng chính (Main Program)

- **Quản lý sản phẩm:**
 - Thêm sản phẩm vào danh sách.
 - Xóa sản phẩm theo `productId`.
 - Hiển thị danh sách sản phẩm.
- **Quản lý đơn hàng:**
 - Tạo đơn hàng mới.
 - Hiển thị danh sách đơn hàng.
- **Báo cáo doanh thu:**
 - Tính tổng doanh thu từ tất cả đơn hàng.

Tiêu chí đánh giá dự án

- **Tính hoàn thiện:** Chương trình phải có đầy đủ các chức năng theo yêu cầu.
- **Tính chính xác:** Chương trình không được lỗi logic.
- **Tính bảo mật dữ liệu:** Sử dụng `private` cho các thuộc tính và `getter/setter` hợp lý.
- **Tổ chức mã nguồn:** Code rõ ràng, dễ đọc, có chú thích hợp lý.
- **Tính hợp tác nhóm:** Mỗi thành viên đóng góp đầy đủ vào dự án.

Lưu ý cho sinh viên

- **Tư duy lập trình hướng đối tượng:** Tổ chức mã nguồn theo các lớp `Product`, `Order` một cách khoa học.
- **Kiểm tra đầu vào:** Đảm bảo không có lỗi nhập liệu, như giá sản phẩm không được âm, số lượng sản phẩm không âm.
- **Tối ưu hóa mã nguồn:** Dùng **`ArrayList`** để lưu danh sách sản phẩm và đơn hàng.
- **Hợp tác hiệu quả:** Chia sẻ code, kiểm thử chương trình trước khi nộp.