

Bài Giảng: Lập Trình Hướng Đối Tượng Buổi 1

1. Giới thiệu học phần:

- **Mục tiêu:** Cung cấp kiến thức về lập trình hướng đối tượng (OOP) với Java.
- **Các nội dung chính:**
 - Căn bản về Java.
 - Class và Object.
 - Kỹ thuật kế thừa.
 - Kỹ thuật đóng gói.
 - Kỹ thuật trừu tượng.
 - Kỹ thuật đa hình.
 - Phân tích, thiết kế và lập trình ứng dụng hoàn chỉnh bằng Java.
 - Làm việc nhóm, tìm kiếm và đọc hiểu tài liệu, phân tích và thiết kế chương trình.

2. Tổng quan lập trình hướng đối tượng:

- **Lập trình hướng cấu trúc:** Chương trình chính được chia thành các chương trình con, mỗi chương trình con thực hiện một nhiệm vụ xác định.
- **Lập trình hướng đối tượng:** Tập trung vào đối tượng và các thuộc tính, phương thức của đối tượng.

3. So sánh lập trình truyền thống và lập trình hướng đối tượng:

- **Phương pháp mô hình:**
 - Lập trình truyền thống: Từ tổng quan đến chi tiết.
 - Lập trình hướng đối tượng: Từ cụ thể đến trừu tượng.
- **Đặc trưng đóng gói:**
 - Lập trình truyền thống: Cấu trúc dữ liệu và giải thuật có mối quan hệ chặt chẽ.
 - Lập trình hướng đối tượng: Đóng gói dữ liệu, sử dụng lại mã nguồn.
- **Ưu điểm:**
 - Lập trình truyền thống: Giải thuật rõ ràng, dễ theo dõi.
 - Lập trình hướng đối tượng: Bảo vệ dữ liệu, tái sử dụng mã nguồn.
- **Nhược điểm:**

- Lập trình truyền thống: Không bảo vệ dữ liệu, phụ thuộc vào cấu trúc dữ liệu.
- Lập trình hướng đối tượng: Phức tạp hơn.

4. Xu hướng lập trình và cài đặt môi trường:

- **Xu hướng phát triển của lập trình hướng đối tượng:**
 - **Lập trình hướng thành phần:** Xây dựng các thành phần độc lập tương đối, có thể ghép lại với nhau để tạo thành phần mềm đáp ứng yêu cầu.
 - **Lập trình hướng agent:** Các agent là các thành phần tự chủ, có khả năng liên lạc, phối hợp hoặc cạnh tranh để hoàn thành nhiệm vụ.
 - **Lập trình hướng aspect:** Đóng gói theo luồng công việc hoặc khía cạnh của vấn đề, thực hiện các nhiệm vụ liên tiếp nhau.

5. Bài tập thực hành:

- **Bài tập cơ bản:**
 - Viết chương trình Hello World bằng Java.
 - Tạo lớp và đối tượng đơn giản.
 - Tính toán diện tích và chu vi các hình học cơ bản.
 - Quản lý thông tin sinh viên.
- **Bài tập trung bình:**
 - Mô phỏng quản lý thư viện.
 - Quản lý lịch làm việc.
- **Bài tập nâng cao:**
 - Xây dựng các ứng dụng phức tạp hơn để áp dụng kiến thức đã học.
- **Chuẩn Đầu Ra:**
- **Kiến thức:** Sinh viên nắm vững các khái niệm và kỹ thuật lập trình hướng đối tượng.
- **Kỹ năng:** Phân tích, thiết kế và lập trình ứng dụng Java hoàn chỉnh; làm việc nhóm và đọc hiểu tài liệu chuyên ngành.
- **Thái độ:** Chủ động tìm kiếm, học hỏi và vận dụng kiến thức để giải quyết các vấn đề thực tế.
- **Đánh Giá Học Phần:**
- **Đánh giá thường xuyên (25%):** Làm và nộp bài thực hành hằng tuần.

- **Đánh giá giữa kỳ (15%):** Kiểm tra vào tuần thứ 6.
- **Đánh giá chuyên cần (10%):** Đọc tài liệu, bình luận, làm bài quiz, tham gia học đầy đủ.
- **Thi cuối kỳ (50%):** Thi thực hành, do phòng đào tạo tổ chức.

Bài 1. Tạo Chương Trình Tính Toán Diện Tích Và Chu Vi Của Các Hình Học Cơ Bản

Mục tiêu:

Sinh viên sẽ tạo một chương trình tính toán diện tích và chu vi cho các hình học cơ bản như hình chữ nhật, hình tròn và hình tam giác bằng ngôn ngữ lập trình Java.

Yêu cầu:

1. Xây dựng các lớp cho các hình học cơ bản:

- **Lớp Rectangle (Hình Chữ Nhật):**
 - **Thuộc tính:** width (chiều rộng), height (chiều cao).
 - **Phương thức:** calculateArea(), calculatePerimeter().
- **Lớp Circle (Hình Tròn):**
 - **Thuộc tính:** radius (bán kính).
 - **Phương thức:** calculateArea(), calculatePerimeter().
- **Lớp Triangle (Hình Tam Giác):**
 - **Thuộc tính:** base (cạnh đáy), height (chiều cao), sideA, sideB, sideC (các cạnh của tam giác).
 - **Phương thức:** calculateArea(), calculatePerimeter().

Mã nguồn Java cho bài toán:

```
// Lớp Hình Chữ Nhật
class Rectangle {
    private double width;
    private double height;

    public Rectangle(double width, double height) {
        this.width = width;
        this.height = height;
    }

    public double calculateArea() {
        return width * height;
    }

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

```
}
```

```
// Lớp Hình Tròn
```

```
class Circle {
```

```
    private double radius;
```

```
    public Circle(double radius) {  
        this.radius = radius;  
    }
```

```
    public double calculateArea() {  
        return Math.PI * radius * radius;  
    }
```

```
    public double calculatePerimeter() {  
        return 2 * Math.PI * radius;  
    }
```

```
}
```

```
// Lớp Hình Tam Giác
```

```
class Triangle {
```

```
    private double base;  
    private double height;  
    private double sideA;  
    private double sideB;  
    private double sideC;
```

```
    public Triangle(double base, double height, double sideA, double sideB, double  
sideC) {  
        this.base = base;  
        this.height = height;  
        this.sideA = sideA;  
        this.sideB = sideB;  
        this.sideC = sideC;  
    }
```

```
    public double calculateArea() {  
        return 0.5 * base * height;  
    }
```

```
    public double calculatePerimeter() {  
        return sideA + sideB + sideC;  
    }
```

```
}  
}
```

// Lớp chính để chạy chương trình

public class Main {

public static void main(String[] args) {

// Tạo đối tượng Hình Chữ Nhật

Rectangle rectangle = new Rectangle(5.0, 3.0);

System.out.println("Rectangle Area: " + rectangle.calculateArea());

System.out.println("Rectangle Perimeter: " + rectangle.calculatePerimeter());

// Tạo đối tượng Hình Tròn

Circle circle = new Circle(4.0);

System.out.println("Circle Area: " + circle.calculateArea());

System.out.println("Circle Perimeter: " + circle.calculatePerimeter());

// Tạo đối tượng Hình Tam Giác

Triangle triangle = new Triangle(3.0, 4.0, 3.0, 4.0, 5.0);

System.out.println("Triangle Area: " + triangle.calculateArea());

System.out.println("Triangle Perimeter: " + triangle.calculatePerimeter());

```
}  
}
```

Giải thích:

- **Lớp Rectangle:** Định nghĩa lớp Rectangle với các thuộc tính width và height. Phương thức calculateArea tính diện tích và phương thức calculatePerimeter tính chu vi.
- **Lớp Circle:** Định nghĩa lớp Circle với thuộc tính radius. Phương thức calculateArea tính diện tích và phương thức calculatePerimeter tính chu vi.
- **Lớp Triangle:** Định nghĩa lớp Triangle với các thuộc tính base, height, sideA, sideB, và sideC. Phương thức calculateArea tính diện tích và phương thức calculatePerimeter tính chu vi.
- **Lớp chính Main:** Chứa phương thức main, là điểm bắt đầu của chương trình, tạo các đối tượng Rectangle, Circle, và Triangle và gọi các phương thức để hiển thị diện tích và chu vi của từng hình.

Bài 2. Bài Tập Ứng Dụng Thực Tiễn: Quản Lý Nhà Hàng

Mục tiêu:

Sinh viên sẽ hiểu và áp dụng các khái niệm cơ bản của lập trình hướng đối tượng thông qua việc xây dựng các lớp và các chức năng chính trong một ứng dụng quản lý nhà hàng.

Yêu cầu:

1. **Xây dựng các lớp chính:**

- **Lớp Customer (Khách Hàng):**
 - **Thuộc tính:** name (tên), phoneNumber (số điện thoại).
 - **Phương thức:** getName(), getPhoneNumber(), setName(), setPhoneNumber().
 - **Lớp MenuItem (Món Ăn):**
 - **Thuộc tính:** name (tên món), price (giá), description (mô tả món ăn).
 - **Phương thức:** getName(), getPrice(), getDescription(), setName(), setPrice(), setDescription().
 - **Lớp Order (Đơn Đặt Hàng):**
 - **Thuộc tính:** orderId (mã đơn hàng), items (danh sách món ăn), totalAmount (tổng giá trị đơn hàng).
 - **Phương thức:** getOrderId(), getItems(), getTotalAmount(), addItem(), removeItem().
- 2. Chức năng chính:**
- **Quản lý khách hàng:**
 - Thêm khách hàng mới.
 - Chỉnh sửa thông tin khách hàng.
 - **Quản lý món ăn:**
 - Thêm món ăn mới.
 - Chỉnh sửa thông tin món ăn.
 - **Quản lý đơn đặt hàng:**
 - Tạo đơn đặt hàng mới.
 - Thêm và xóa món ăn trong đơn đặt hàng.
 - Tính tổng giá trị đơn hàng.

Bước thực hiện:

- 1. Phân tích yêu cầu:**
 - Xác định các yêu cầu cơ bản của ứng dụng.
 - Thiết kế sơ đồ lớp và mối quan hệ giữa các lớp.
- 2. Thiết kế lớp:**
 - Xây dựng các lớp Customer, MenuItem, và Order với các thuộc tính và phương thức tương ứng.
- 3. Cài đặt và kiểm thử:**
 - Viết mã nguồn cho các lớp và phương thức đã thiết kế.
 - Kiểm thử từng phần để đảm bảo hoạt động đúng chức năng.

Code chương trình bài trên:

```
// Lớp Khách Hàng
class Customer {
    private String name;
    private String phoneNumber;

    public Customer(String name, String phoneNumber) {
```

```
this.name = name;
this.phoneNumber = phoneNumber;
}

// Phương thức getter và setter
public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public String getPhoneNumber() {
    return phoneNumber;
}

public void setPhoneNumber(String phoneNumber) {
    this.phoneNumber = phoneNumber;
}
}
```

```
// Lớp Món Ăn
class MenuItem {
    private String name;
    private double price;
    private String description;

    public MenuItem(String name, double price, String description) {
        this.name = name;
        this.price = price;
        this.description = description;
    }

    // Phương thức getter và setter
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

```

    public double getPrice() {
        return price;
    }

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

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }
}

```

```

// Lớp Đơn Đặt Hàng
import java.util.ArrayList;
import java.util.List;

class Order {
    private String orderId;
    private List<MenuItem> items;
    private double totalAmount;

    public Order(String orderId) {
        this.orderId = orderId;
        this.items = new ArrayList<>();
        this.totalAmount = 0.0;
    }

    // Phương thức thêm món ăn vào đơn đặt hàng
    public void addItem(MenuItem item) {
        items.add(item);
        totalAmount += item.getPrice();
    }

    // Phương thức xóa món ăn khỏi đơn đặt hàng
    public void removeItem(MenuItem item) {
        items.remove(item);
        totalAmount -= item.getPrice();
    }
}

```



```

    }

    // Phương thức getter
    public String getOrderId() {
        return orderId;
    }

    public List<MenuItem> getItems() {
        return items;
    }

    public double getTotalAmount() {
        return totalAmount;
    }
}

```

```

public class RestaurantManagement {
    public static void main(String[] args) {
        // Tạo khách hàng
        Customer customer = new Customer("Nguyen Van A", "0900123456");

        // Tạo món ăn
        MenuItem item1 = new MenuItem("Pho", 40000, "Pho bo tai");
        MenuItem item2 = new MenuItem("Bun Cha", 45000, "Bun cha Hanoi");

        // Tạo đơn đặt hàng và thêm món ăn
        Order order = new Order("ORD1234");
        order.addItem(item1);
        order.addItem(item2);

        // Hiển thị thông tin đơn đặt hàng
        System.out.println("Order ID: " + order.getOrderId());
        System.out.println("Customer: " + customer.getName() + " - " +
customer.getPhoneNumber());
        System.out.println("Total Amount: " + order.getTotalAmount() + " VND");
        for (MenuItem item : order.getItems()) {
            System.out.println("Item: " + item.getName() + " - " + item.getPrice() + "
VND");
        }
    }
}

```

Bài 3. Bài Tập: Chương Trình Quản Lý Thông Tin Sinh Viên

Mục tiêu:

Sinh viên sẽ viết một chương trình bằng ngôn ngữ lập trình Java để quản lý thông tin sinh viên, bao gồm các thuộc tính như tên, tuổi, và mã sinh viên.

Yêu cầu:

1. Xây dựng lớp Student (Sinh Viên):

o Thuộc tính:

- name (tên, dạng chuỗi)
- age (tuổi, dạng số nguyên)
- studentId (mã sinh viên, dạng chuỗi)

o Constructor: Khởi tạo đối tượng Student với các giá trị ban đầu cho các thuộc tính.

o Phương thức getter và setter: Truy xuất và thay đổi giá trị các thuộc tính.

o Phương thức displayInfo: Hiển thị thông tin sinh viên.

2. Tạo lớp chính (Main) để chạy chương trình:

o Trong phương thức main:

- Tạo đối tượng Student với các giá trị cụ thể.
- Gọi phương thức displayInfo để hiển thị thông tin của đối tượng Student.

Mã nguồn Java cho bài toán:

```
// Lớp Sinh Viên
class Student {
    // Các thuộc tính của lớp Student
    private String name;
    private int age;
    private String studentId;

    // Constructor để khởi tạo đối tượng Student
    public Student(String name, int age, String studentId) {
        this.name = name;
        this.age = age;
        this.studentId = studentId;
    }

    // Các phương thức getter và setter
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

```

public int getAge() {
    return age;
}

public void setAge(int age) {
    this.age = age;
}

public String getStudentId() {
    return studentId;
}

public void setStudentId(String studentId) {
    this.studentId = studentId;
}

// Phương thức để hiển thị thông tin sinh viên
public void displayInfo() {
    System.out.println("Name: " + name);
    System.out.println("Age: " + age);
    System.out.println("Student ID: " + studentId);
}
}

```

```

// Lớp chính để chạy chương trình
public class Main {
    public static void main(String[] args) {
        // Tạo đối tượng Student
        Student student = new Student("Nguyen Van A", 20, "SV123456");

        // Hiển thị thông tin của đối tượng Student
        student.displayInfo();
    }
}

```


BÀI TẬP ÔN TẬP KỸ THUẬT LẬP TRÌNH (JAVA)

Mục tiêu:

- Ôn tập lại các khái niệm lập trình cơ bản: biến, toán tử, vòng lặp, điều kiện, phương thức.
- Làm quen với cách tổ chức mã nguồn theo hướng cấu trúc trước khi chuyển sang hướng đối tượng.

- Rèn luyện tư duy logic và kỹ năng viết mã trong Java.
-

PHẦN 1: CÂU HỎI TỰ LUẬN NGẮN


 **Yêu cầu:** Viết câu trả lời ngắn gọn, súc tích cho các câu hỏi sau:

1. Java là ngôn ngữ biên dịch hay thông dịch? Giải thích cách Java thực thi một chương trình.
 2. So sánh `==` và `.equals()` khi sử dụng với kiểu String trong Java.
 3. Giải thích sự khác nhau giữa `break` và `continue` trong vòng lặp.
 4. Tại sao nên sử dụng phương thức thay vì viết mã lặp lại nhiều lần?
 5. Viết đoạn mã khai báo và sử dụng mảng một chiều trong Java.
-

PHẦN 2: BÀI TẬP LẬP TRÌNH CƠ BẢN

 **Yêu cầu:** Viết chương trình Java để giải quyết các bài toán sau.


Bài 1: Kiểm tra số nguyên tố

 **Mô tả:** Viết chương trình kiểm tra xem một số nguyên n có phải là số nguyên tố không.

 **Gợi ý:**

- Nhập số từ bàn phím.
 - Số nguyên tố là số lớn hơn 1 và chỉ chia hết cho 1 và chính nó.
-


Bài 2: Tính tổng các số chẵn trong mảng

 **Mô tả:** Viết chương trình nhập một mảng số nguyên n phần tử và tính tổng các số chẵn trong mảng.

 **Gợi ý:**

- Nhập số phần tử mảng.
 - Nhập từng phần tử vào mảng.
 - Duyệt qua mảng để tính tổng các số chẵn.
-


Bài 3: Đảo ngược chuỗi

 **Mô tả:** Viết chương trình nhập vào một chuỗi và in ra chuỗi đảo ngược.

 **Gợi ý:**

- Dùng `charAt()` để lấy từng ký tự từ cuối lên đầu.
 - Có thể dùng `StringBuilder.reverse()`.
-

PHẦN 3: BÀI TẬP NÂNG CAO

 **Yêu cầu:** Sinh viên áp dụng kiến thức đã học để hoàn thành bài tập phức tạp hơn.

Bài 4: Quản lý sinh viên (dùng mảng)

💡 **Mô tả:** Viết chương trình quản lý thông tin sinh viên bao gồm: **họ tên, mã số, điểm trung bình.**

🔍 **Yêu cầu:**

- Nhập danh sách sinh viên từ bàn phím (sử dụng mảng).
 - Hiển thị danh sách sinh viên.
 - Tìm kiếm sinh viên theo mã số.
-

Bài 5: Mô phỏng ATM (dùng vòng lặp)

💡 **Mô tả:** Viết chương trình mô phỏng máy ATM, cho phép người dùng thực hiện các thao tác:

1. Kiểm tra số dư.
2. Nạp tiền vào tài khoản.
3. Rút tiền (nếu số dư đủ).
4. Thoát chương trình.

🔍 **Gợi ý:**

- Dùng while để lặp liên tục đến khi người dùng chọn "Thoát".
- Kiểm tra số dư trước khi rút tiền.