Session 06:

Tổng quan về OOP, Class và Object

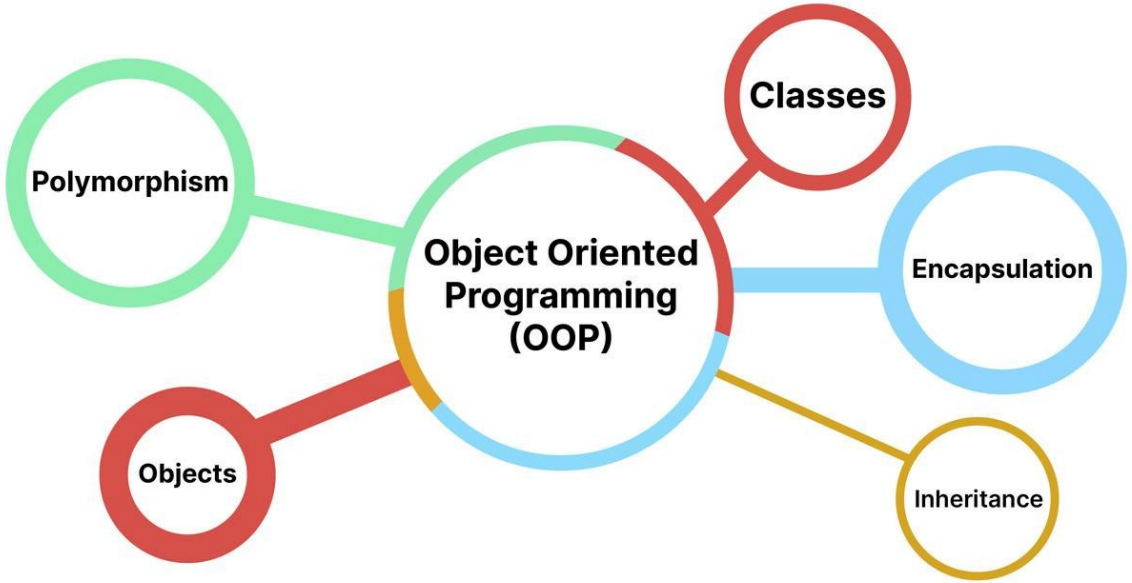
NỘI DUNG

1. Tổng quan về OOP, Class và Object
2. Thuộc tính, phương thức, constructor, this
3. Các đặc tả truy xuất, đóng gói, getter, setter
4. Khai báo và khởi tạo đối tượng
5. Truy xuất thuộc tính, phương thức của đối tượng

1. Tổng quan về OOP, Class và Object

OOP:

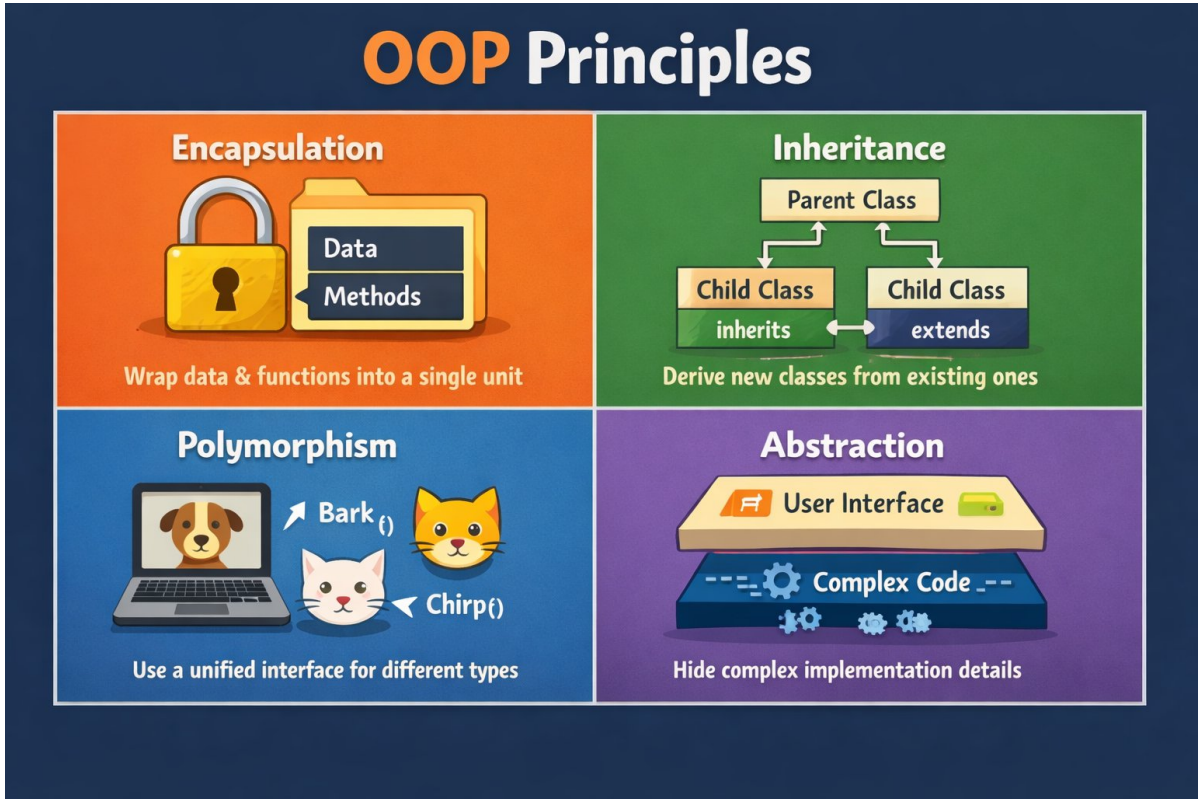
- Là phương pháp lập trình dựa trên đối tượng (Object).
- Mỗi đối tượng gồm:
  - **Thuộc tính** (**Attribute/Property**): Dữ liệu, trạng thái.
  - **Phương thức** (**Method**): Hành vi, chức năng.



1. Tổng quan về OOP, Class và Object

4 tính chất của OOP:

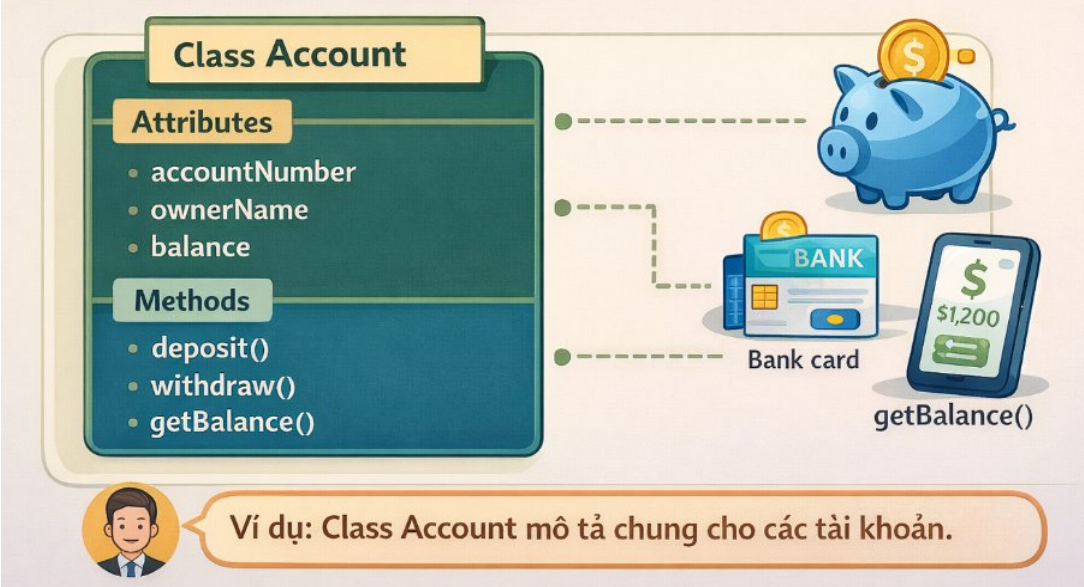
- Đóng gói (Encapsulation)
- Kế thừa (Inheritance)
- Đa hình (Polymorphism)
- Trừu tượng (Abstraction)

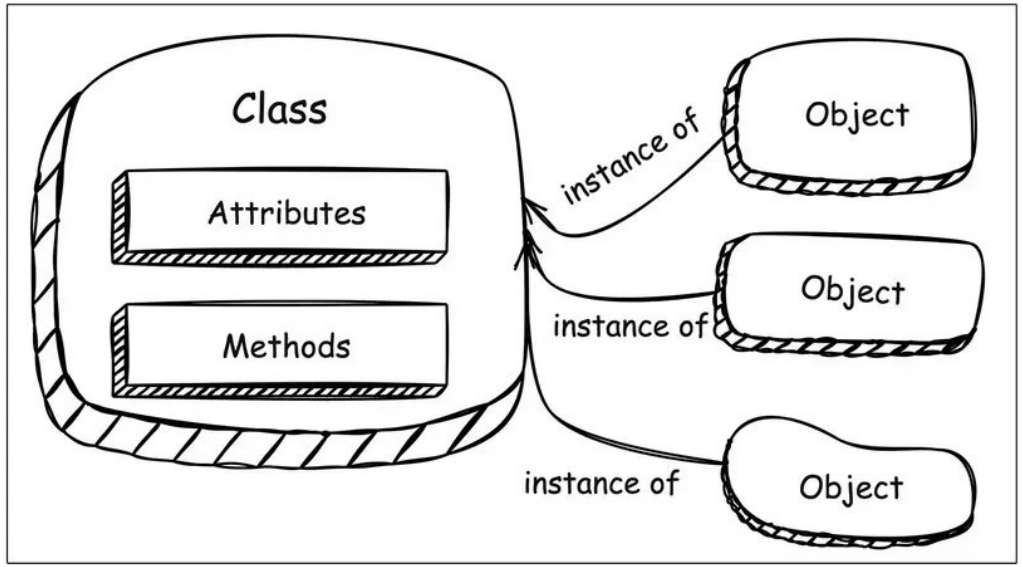


1. Tổng quan về OOP, Class và Object

Class (Lớp)

- Là một **bản thiết kế** (**blueprint**) hoặc **khuôn mẫu** (**template**) được sử dụng để định nghĩa các **thuộc tính**, **phương thức** cho các đối tượng.
- Để **tạo ra lớp**, ta **sử dụng từ khóa class**

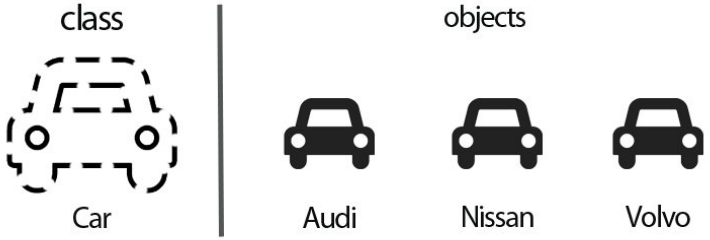


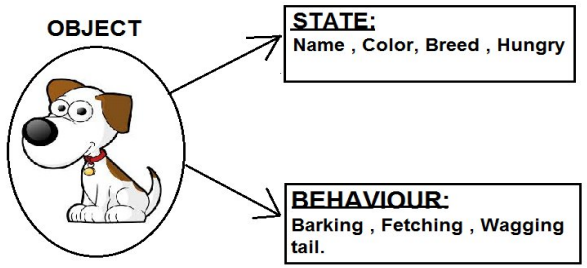


1. Tổng quan về OOP, Class và Object

Object (Đối tượng)

- Là **một thể hiện cụ thể của lớp**.
- Được **tạo ra từ lớp bằng từ khóa new**
- Mỗi **object** có bộ nhớ riêng, lưu trữ các thuộc tính riêng.





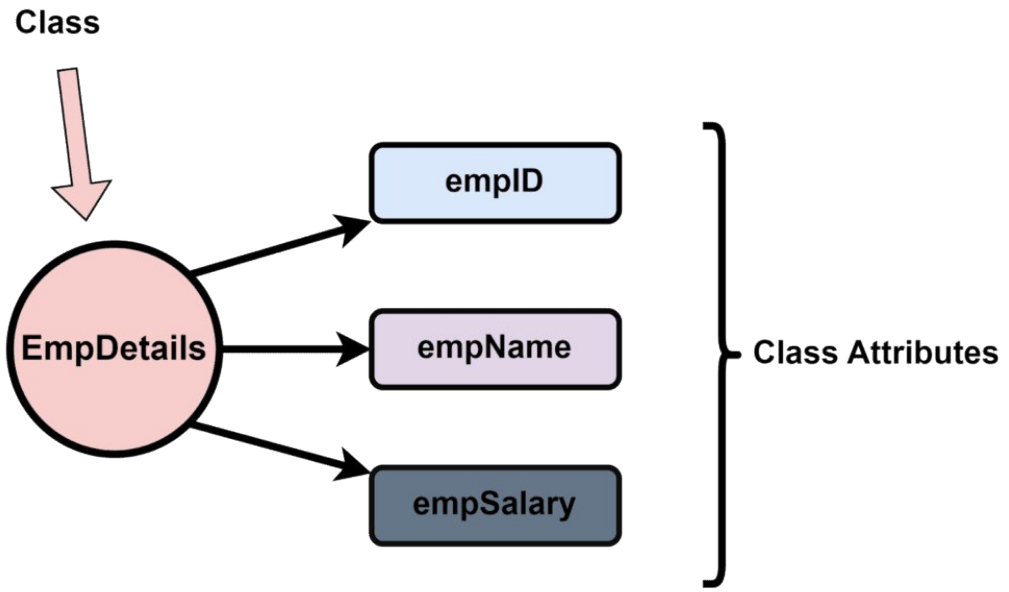

```java
// Khai báo lớp (Khuôn mẫu)
class Dog {
    String breed; // Trạng thái
    int age;

    void bark() { // Hành vi
        System.out.println("Woof! Woof!");
    }
}

// Tạo đối tượng (Thực thể cụ thể)
public class Main {
    public static void main(String[] args) {
        Dog myDog = new Dog(); // myDog là một Object
        myDog.breed = "BullDog";
        myDog.age = 3;
        myDog.bark(); // Gọi hành vi
    }
}
```

2. Thuộc tính, phương thức, constructor, this

Thuộc tính (Attribute/Property)

- Là các biến lưu trữ trạng thái của đối tượng.
- Ví dụ trong class Account:

```
1  public class Account {
2      public int id;
3      public String email;
4      public String userName;
5      public String fullName;
6  }
```



2. Thuộc tính, phương thức, constructor, this

Phương thức (Method)

- Là các hàm/thủ tục mô tả hành vi của đối tượng.

```
modifier returnType methodName(parameterList) {
  // Thân phương thức (Các câu lệnh xử lý)
}
```

Thành phần	Ý nghĩa	Ví dụ
Modifier	Phạm vi truy cập (Ai được phép dùng nó?)	public, private, protected
Return Type	Kiểu dữ liệu trả về sau khi làm xong.	int, String, hoặc void (nếu không trả về gì).
Method Name	Tên của hành động (thường dùng động từ).	run, calculateTotal, getName
Parameter List	Các nguyên liệu đầu vào (nằm trong ngoặc đơn).	(int a, int b)
Method Body	Nơi viết code để giải quyết vấn đề.	{ return a + b; }

2. Thuộc tính, phương thức, constructor, this

Constructor là gì?

- Là **phương thức đặc biệt**, được **gọi tự động khi tạo đối tượng**.
- **Mục đích:** Khởi tạo giá trị ban đầu cho đối tượng.
- **Quy tắc:**
  - Tên constructor phải trùng với tên lớp.
  - Không có kiểu trả về (kể cả void).

2. Thuộc tính, phương thức, constructor, this

Constructor mặc định và có tham số

Constructor mặc định: Không có tham số, giá trị mặc định (0, null).

```
1  public Account() {
2      // Khởi tạo mặc định
3  }
```

Constructor có tham số: Truyền giá trị cụ thể khi khởi tạo.

```
1  public Account(int id, String email, String userName, String fullName) {
2      this.id = id;
3      this.email = email;
4      this.userName = userName;
5      this.fullName = fullName;
6  }
```

2. Thuộc tính, phương thức, constructor, this

Constructor Overloading

- Một lớp có nhiều constructor, khác nhau về số lượng và kiểu tham số.

```
1  public Account() { ... }
2  public Account(int id, String email) { ... }
3  public Account(int id, String email, String userName, String fullName) { ... }
```

2. Thuộc tính, phương thức, constructor, this

Từ khóa "this":

- Tham chiếu đến đối tượng hiện tại.
- Dùng để phân biệt thuộc tính và tham số khi trùng tên.

```
1  public void setEmail(String email) {
2      this.email = email;   // this.email là thuộc tính, email là tham số
3  }
```

3. Các Đặc Tả Truy Xuất, Đóng Gói, Getter, Setter

Access Modifier (Phạm vi truy cập)

- Điều khiển khả năng truy cập đến thuộc tính, phương thức, constructor.
- Bảng tóm tắt:

Modifier	Class	Package	Subclass	Global
Public	Yes	Yes	Yes	Yes
Protected	Yes	Yes	Yes	No
Default	Yes	Yes	No	No
Private	Yes	No	No	No

3. Các Đặc Tả Truy Xuất, Đóng Gói, Getter, Setter

Tính đóng gói (Encapsulation)

- Ẩn giấu thông tin bên trong đối tượng, chỉ hiển thị ra bên ngoài những gì cần thiết.
- Cách thực hiện:
  - Khai báo thuộc tính là private.
  - Cung cấp phương thức public (getter/setter) để truy cập và sửa đổi.

```java
public class Account {
    // 1. Thuộc tính để private: Giấu kín số dư
    private double balance;

    // 2. Phương thức Setter (Người gác cổng - Kiểm soát dữ liệu vào)
    public void setBalance(double amount) {
        if (amount >= 0) { // Kiểm tra điều kiện: Không được nạp tiền âm
            this.balance = amount;
        } else {
            System.out.println("Lỗi: Số dư không thể âm!");
        }
    }

    // 3. Phương thức Getter (Hiển thị dữ liệu ra ngoài một cách an toàn)
    public double getBalance() {
        return this.balance;
    }
}
```

3. Các Đặc Tả Truy Xuất, Đóng Gói, Getter, Setter

Getter và Setter

- **Getter:** Lấy giá trị thuộc tính private.
- **Setter:** Thiết lập giá trị thuộc tính private (có thể kiểm tra tính hợp lệ).

```java
public String getEmail() {
    return email;
}
public void setEmail(String email) {
    // Có thể kiểm tra email hợp lệ trước khi gán
    if (email.contains("@")) {
        this.email = email;
    } else {
        System.out.println("Email không hợp lệ!");
    }
}
```

3. Các Đặc Tả Truy Xuất, Đóng Gói, Getter, Setter

Lợi ích của đóng gói

- Kiểm soát truy cập: Đảm bảo dữ liệu không bị thay đổi trực tiếp từ bên ngoài.
- Dễ dàng thay đổi logic bên trong mà không ảnh hưởng đến code sử dụng.
- Ví dụ: Thay đổi kiểm tra email trong setter mà không cần sửa code nơi gọi.

```java
public void setEmail(String email) {
    if (!email.isEmpty()) {
        this.email = email;
    }
}
```

```java
public void setEmail(String email) {
    if (email.contains("@") && email.endsWith(".com")) {
        this.email = email;
    } else {
        System.out.println("Email không hợp lệ!");
    }
}
```

4. Khai Báo Và Khởi Tạo Đối Tượng

Khai báo đối tượng

- **Cú pháp:** ClassName objectName;
- **Ví dụ:** Account acc1; (chỉ khai báo, chưa có đối tượng)
  - Account a1 = **new** Account();
  - Account a2 = a1;

Khởi tạo đối tượng với ”new”

- Cú pháp: objectName = **new** Constructor();
- Kết hợp: ClassName objectName = **new** Constructor();

```
1  Account acc1 = new Account();  // Gọi constructor mặc định
2  Account acc2 = new Account(1, "demo@rikkei.com", "user", "Name"); // Gọi constructor có tham số
```

4. Khai Báo Và Khởi Tạo Đối Tượng

Ví dụ khởi tạo với các constructor

```
1  // Tạo đối tượng với constructor mặc định
2  Account acc1 = new Account();
3  acc1.setId(1);
4  acc1.setEmail("demo@rikkei.com.vn");
5
6  // Tạo đối tượng với constructor có tham số
7  Account acc2 = new Account(2, "email2", "username2", "fullname2");
```

5. Truy xuất thuộc tính, phương thức của đối tượng

Truy xuất thuộc tính

- Trong Java, dấu chấm (.) giống như một chiếc chìa khóa để mở các cánh cửa bên trong đối tượng. Tuy nhiên, việc mở được hay không tùy thuộc vào loại khóa (**Access Modifier**).
- **Trường hợp Public (Ít dùng trong thực tế):** Giống như một cửa hàng không có bảo vệ, ai cũng có thể vào lấy đồ hoặc thay đổi đồ.

```
1  Account acc1 = new Account();
2  acc1.id = 1;  // Gán giá trị
3  System.out.println(acc1.id); // Lấy giá trị
```

- **Trường hợp Private (Đóng gói):** Giống như một ngân hàng. Bạn không thể tự vào kho lấy tiền, bạn phải gặp nhân viên (Getter/Setter) để thực hiện yêu cầu.

```
1  acc1.setId(1);
2  System.out.println(acc1.getId());
```

5. Truy xuất thuộc tính, phương thức của đối tượng

Gọi phương thức

- Phương thức là các hành động mà đối tượng có thể thực hiện. Cú pháp rất đơn giản: **tên_đối_tượng.tên_phương_thức**(tham_số);
- **Setter/Getter:** Dùng để cập nhật và lấy thông tin.
- **Phương thức hành động:** Thực hiện một chức năng cụ thể

```
1  acc1.setEmail("new@rikkei.com");          // Gọi setter
2  System.out.println(acc1.getEmail());    // Gọi getter
3  acc1.goVTI();                           // Gọi phương thức thông thường
```

TỔNG KẾT

- ❑ Nắm được khái niệm OOP, class, object.
- ❑ Hiểu và sử dụng được constructor, từ khóa this.
- ❑ Hiểu access modifier và tính đóng gói.
- ❑ Biết cách khai báo, khởi tạo, truy xuất đối tượng.
- ❑ Sử dụng được truy xuất thuộc tính, phương thức đối tượng.

RIKEI Academy®
where the dream begins

KẾT THÚC

HỌC VIỆN ĐÀO TẠO LẬP TRÌNH CHẤT LƯỢNG NHẬT BẢN