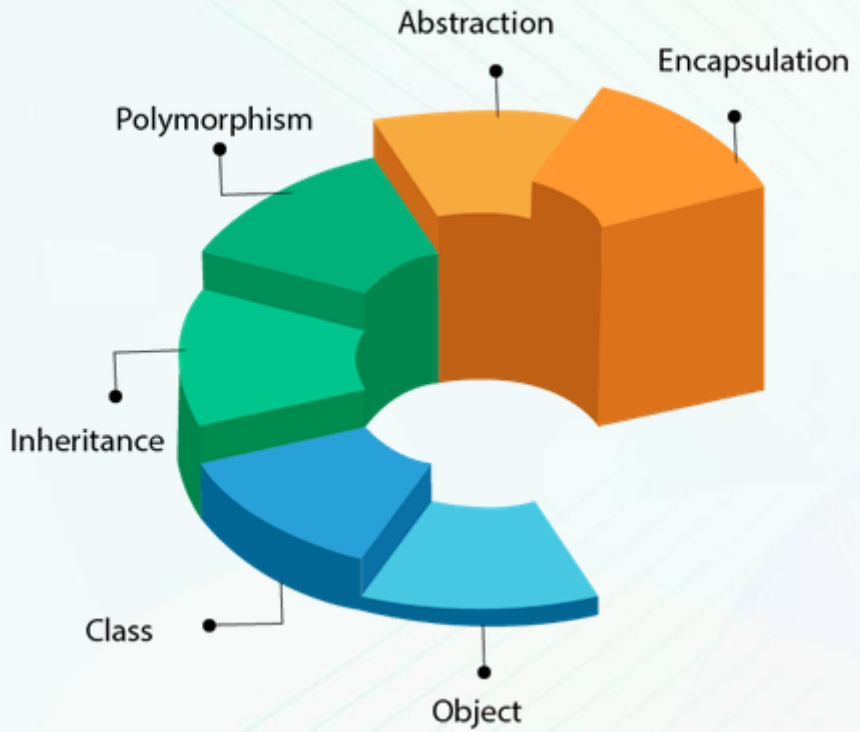
FUNX
Learn with Mentors

Được tạo bởi Nguyễn Quang Vinh (vinhnq1)
Lần chỉnh sửa cuối - 31/01/2025

Chương 1
Khái quát về lập trình hướng đối tượng



Một số phương pháp lập trình

- **Lập trình tuyến tính:** Là phương pháp lập trình đơn giản, thực hiện các lệnh theo thứ tự từ trên xuống dưới.
  - Ưu điểm: Dễ hiểu, dễ viết.
  - Nhược điểm: Khó bảo trì, không thể tái sử dụng mã.
- **Lập trình hướng cấu trúc:** Chương trình được chia thành các hàm (chương trình con) để thực hiện các nhiệm vụ cụ thể.
  - Ưu điểm: Dễ bảo trì, dễ hiểu hơn so với lập trình tuyến tính.
  - Nhược điểm: Khó quản lý dữ liệu dùng chung, khó mô tả hệ thống phức tạp.
- **Lập trình hướng đối tượng (OOP):** Lập trình hướng đối tượng là phương pháp lập trình dựa trên việc tạo ra các đối tượng và tương tác giữa chúng.
  - Ưu điểm: Dễ bảo trì, tái sử dụng mã, mô tả hệ thống phức tạp tốt hơn.
  - Nhược điểm: Phức tạp hơn so với lập trình tuyến tính và hướng cấu trúc.

Ví dụ 1 – Lập trình tuyến tính

```cpp
#include <iostream>
using namespace std;

int main()
{
    int a = 5;
    int b = 10;
    int sum = a + b;
    cout << "Sum: " << sum << endl;
    return 0;
}
```

Ví dụ 2 – Lập trình hướng thủ tục

```cpp
#include <iostream>
using namespace std;

void printSum(int a, int b)
{
    int sum = a + b;
    cout << "Sum: " << sum << endl;
}

int main()
{
    int a = 5;
    int b = 10;
    printSum(a, b);
    return 0;
}
```

Ví dụ 3 – Lập trình hướng đối tượng

```
class Calculator {
    int add(int a, int b) {
        return a + b;
    }
}

public class Main {
    public static void main(String[] args) {
        Calculator calc = new Calculator();
        int result = calc.add(5, 10);
        System.out.println("Sum: " + result);
    }
}
```

So sánh POP và OOP

POP (Procedural-Oriented Programming):

- Tập trung vào thuật toán.
- Dữ liệu và hàm tách biệt.
- Khó quản lý dữ liệu dùng chung.

OOP (Object-Oriented Programming):

- Tập trung vào dữ liệu và đối tượng.
- Dữ liệu và phương thức được đóng gói trong đối tượng.
- Dễ quản lý và bảo trì.

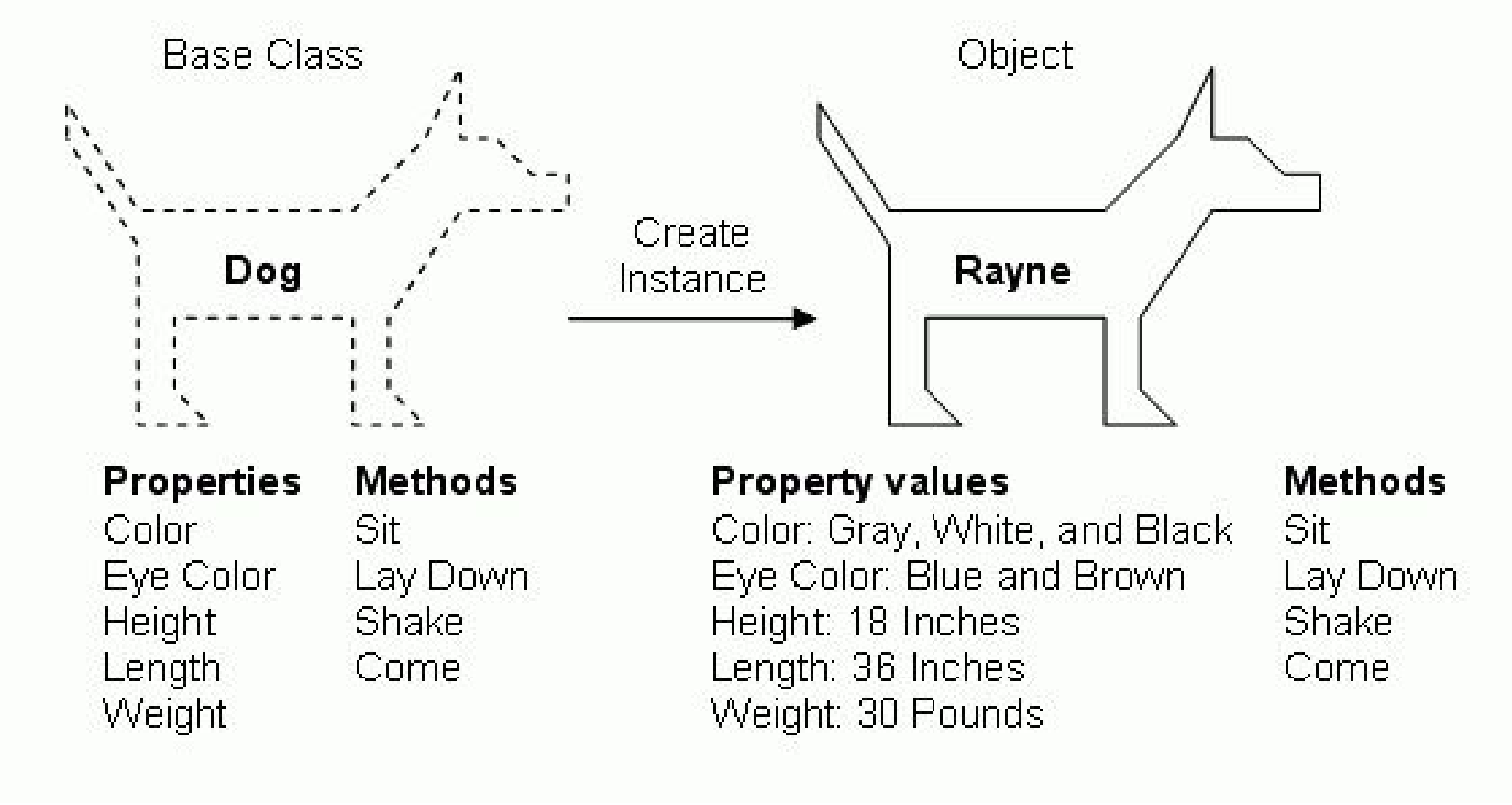
Các khái niệm trong lập trình hướng đối tượng

- Đối tượng (Object)
- Lớp (Class)
- Thông điệp (Message)

Lớp (Class) & Đối tượng (Object)

Lớp là một khuôn mẫu (blueprint) để tạo ra các đối tượng.

Đối tượng là một thực thể trong thế giới thực, có trạng thái (thuộc tính) và hành vi (phương thức).



Tương tác giữa lớp và đối tượng

Nhắc lại: Lớp là khuôn mẫu, đối tượng là thể hiện cụ thể của lớp.

Ví dụ: Từ lớp `Student`, ta có thể tạo ra các đối tượng như `student1`, `student2`.

```
public class Main {
    public static void main(String[] args) {
        Student student1 = new Student();
        student1.name = "Alice";
        student1.age = 20;
        student1.study();

        Student student2 = new Student();
        student2.name = "Bob";
        student2.age = 22;
        student2.attendClass();
    }
}
```

Thông điệp (Message)

- Thông điệp (Message) là một yêu cầu mà một đối tượng gửi đến một đối tượng khác để yêu cầu thực hiện một hành vi (phương thức) cụ thể.
- Trong lập trình, thông điệp thường được thể hiện qua việc gọi một phương thức của một đối tượng.

Các thành phần của thông điệp gồm có:

- **Đối tượng đích:** Đối tượng nhận thông điệp.
- **Phương thức cần thực hiện:** Hành vi mà đối tượng đích cần thực hiện.
- **Tham số (nếu có):** Dữ liệu cần thiết để thực hiện phương thức.

Ví dụ 4 - Thông điệp (Message)

```
class Calculator {
    // Phương thức thực hiện phép cộng
    int add(int a, int b) {
        return a + b;
    }
}
class Student {
    void doHomework() {
        Calculator calc = new Calculator(); // Tạo đối tượng Calculator
        int result = calc.add(5, 10); // Gửi thông điệp "add" đến đối tượng calc
        System.out.println("Result: " + result);
    }
}
public class Main {
    public static void main(String[] args) {
        Student student = new Student(); // Tạo đối tượng Student
        student.doHomework(); // Gửi thông điệp "doHomework" đến đối tượng student
    }
}
```

Các tính chất trong lập trình hướng đối tượng

- Tính trừu tượng (Abstraction)
- Tính đóng gói (Encapsulation)
- Tính đa hình (Polymorphism)
- Tính kế thừa (Inheritance)

Tính trừu tượng (Abstraction)

- Chỉ hiển thị các thông tin cần thiết và ẩn đi các chi tiết phức tạp.
- **Ví dụ:** Khi sử dụng điện thoại, người dùng chỉ cần biết cách gọi điện, không cần biết cách điện thoại hoạt động bên trong.

```
abstract class Animal {
    abstract void makeSound();
}

class Dog extends Animal {
    void makeSound() {
        System.out.println("Woof!");
    }
}
```

Tính đóng gói (Encapsulation)

- **Định nghĩa:** Đóng gói dữ liệu và phương thức liên quan vào một đơn vị (lớp), và hạn chế truy cập từ bên ngoài.
- **Ví dụ:** Thuộc tính `balance` trong lớp `BankAccount` được đóng gói và chỉ có thể truy cập thông qua phương thức `getBalance()`.

```
class BankAccount {
    private double balance;

    public double getBalance() {
        return balance;
    }
    public void deposit(double amount) {
        if (amount > 0) {
            balance += amount;
        }
    }
}
```

Tính đa hình (Polymorphism)

- Một phương thức có thể có nhiều hình thức khác nhau tùy vào đối tượng gọi nó.
- **Ví dụ:** Phương thức `makeSound()` của lớp `Animal` có thể được triển khai khác nhau trong lớp `Dog` và `Cat`.

```
class Animal {
    void makeSound() {
        System.out.println("Animal
sound");
    }
}

class Dog extends Animal {
    void makeSound() {
        System.out.println("Woof!");
    }
}
```

```
class Cat extends Animal {
    void makeSound() {
        System.out.println("Meow!");
    }
}
```

Tính kế thừa (Inheritance)

- Cho phép một lớp kế thừa các thuộc tính và phương thức từ một lớp khác.
- **Ví dụ:** Lớp `Dog` và `Cat` kế thừa từ lớp `Animal`.

```
class Animal {
    void eat() {
        System.out.println("Eating ... ");
    }
}

class Dog extends Animal {
    void bark() {
        System.out.println("Woof!");
    }
}
```

Tổng kết

- **Lập trình tuyến tính**: Đơn giản, dễ hiểu nhưng khó bảo trì.
- **Lập trình hướng cấu trúc**: Dễ bảo trì hơn nhưng vẫn khó quản lý dữ liệu dùng chung.
- **Lập trình hướng đối tượng (OOP)**: Dễ bảo trì, tái sử dụng mã, và mô tả hệ thống phức tạp tốt hơn.
- **Các khái niệm trong OOP**: Đối tượng, lớp, thông điệp.
- **Các tính chất trong OOP**: Tính trừu tượng, tính đóng gói, tính đa hình, tính kế thừa.