## PHT C# [05] - LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG (OOP)

### 🎯 KIẾN THỨC CỐT LỐI

| **Khái Niệm** | **Ý Nghĩa** | **Ví Dụ** |
| --- | --- | --- |
| **Lớp (Class)** | Bản thiết kế cho đối tượng | class Student { } |
| **Đối tượng (Object)** | Thể hiện cụ thể của lớp | Student sv1 = new Student(); |
| **Thuộc tính (Property)** | Dữ liệu của đối tượng | sv1.Name, sv1.Age |
| **Phương thức (Method)** | Hành động của đối tượng | sv1.Display(), sv1.Calculate() |
| **Constructor** | Phương thức khởi tạo đối tượng | Student(string name) { ... } |
| **Encapsulation** | Che giấu chi tiết, chỉ mở công khai cần thiết | private, public, protected |
| **Kế thừa (Inheritance)** | Lớp con thừa từ lớp cha | class Dog : Animal { } |
| **Đa hình (Polymorphism)** | Một hành động, nhiều cách thực thi | virtual và override |
| **Giao diện (Interface)** | Hợp đồng quy định phương thức bắt buộc | interface IMovable { } |

### 💡 KHÁM PHÁ KIẾN THỨC

**Câu hỏi tự kiểm tra:**

1. Sự khác biệt giữa lớp và đối tượng?

-Lớp: Là bản thiết kế, khuôn mẫu dùng để tạo ra đối tượng

Không chiếm bộ nhớ khi khai báo

Chỉ mô tả thuộc tính và phương thức

-Đối tượng: Là thực thể được tạo ra từ class

Chiếm bộ nhớ khi được tạo bằng **new**

Có dữ liệu thật và có thể thực thi hành động

1. Constructor là gì? Tại sao cần constructor?

- Constructor là phương thức khởi tạo đối tượng.

- Cần constructor vì:

* Giúp **khởi tạo giá trị mặc định** cho object.
* Đảm bảo object luôn ở trạng thái hợp lệ.
* Giúp code gọn hơn, tránh phải gán giá trị thủ công nhiều lần.

1. private, public, protected khác gì?

-private: Chỉ truy cập trong nội bộ class

-public: Truy cập từ bất cứ đâu.

-protected: Truy cập trong class và các class kế thừa

1. Kế thừa có tác dụng gì?

Kế thừa cho phép **class con sử dụng lại thuộc tính và phương thức của class cha.** Giảm trùng lặp code. Giúp tổ chức hệ thống theo cấu trúc tầng lớp. Cho phép mở rộng chức năng mà không sửa trực tiếp class cha. Kết hợp với polymorphism để tạo hành vi đa hình.

### 📝 VÍ DỤ THỰC HÀNH

**TODO 5.1: [TẠO LỚP CƠ BẢN - STUDENT]**

using System;

namespace BasicClass

{

// TODO: Tạo lớp Student

// Thuộc tính: StudentId, Name, GPA

// Phương thức: Display() (in thông tin)

class Student

    {

        public int StudentId { get; set; }

        public string Name { get; set; }

        public double GPA { get; set; }

        public void Display()

        {

            Console.WriteLine($"ID: {StudentId}, Name: {Name}, GPA: {GPA}");

        }

    }

class Program

{

static void Main(string[] args)

{

// TODO: Tạo 2 đối tượng Student

Student st1 = new Student();

            Student st2 = new Student();

// TODO: Gán giá trị thuộc tính

st1.StudentId = 1;

            st1.Name = "Vuong";

            st1.GPA = 3.5;

            st2.StudentId = 2;

            st2.Name = "Muoi Tam";

            st2.GPA = 3.8;

// TODO: Gọi phương thức Display()

st1.Display();

            st2.Display();

}

}

}

**TODO 5.2: [CONSTRUCTOR - KHỞI TẠO DỮ LIỆU]**

using System;

namespace Constructor

{

// TODO: Tạo lớp Product với Constructor

class Product

    {

        public int ProductId { get; set; }

        public string ProductName { get; set; }

        public double Price { get; set; }

// Constructor nhận tham số: productId, productName, price

public Product(int productId, string productName, double price)

        {

            ProductId = productId;

            ProductName = productName;

            Price = price;

        }

        public void Display()

        {

            Console.WriteLine($"ID: {ProductId}, Name: {ProductName}, Price: {Price}");

        }

    }

class Program

{

static void Main(string[] args)

{

// TODO: Tạo đối tượng Product dùng Constructor

 Product p1 = new Product(101, "PS5", 50000.0);

// TODO: In thông tin sản phẩm

 p1.Display();

}

}

}

**TODO 5.3: [ENCAPSULATION - CHE GIẤU DỮ LIỆU]**

using System;

namespace Encapsulation

{

// TODO: Tạo lớp BankAccount

// - Private field: \_balance (số dư)

class BankAccount

    {

        private double \_balance;

// - Public property: Balance (chỉ đọc, phòng trừy cập trái phép)

public double Balance

        {

            get { return \_balance; }

        }

// - Public method: Deposit(amount) - gửi tiền

public void Deposit(double amount)

        {

            if (amount > 0)

            {

                \_balance += amount;

                Console.WriteLine($"Đã gửi {amount}. Số dư hiện tại: {\_balance}");

            }

            else

            {

                Console.WriteLine("Số tiền gửi phải lớn hơn 0!");

            }

        }

// - Public method: Withdraw(amount) - rút tiền (kiểm tra đủ tiền không)

public void Withdraw(double amount)

        {

            if (amount <= 0)

            {

                Console.WriteLine("Số tiền rút không hợp lệ!");

            }

            else if (amount > \_balance)

            {

                Console.WriteLine("Không đủ tiền để rút!");

            }

            else

            {

                \_balance -= amount;

                Console.WriteLine($"Đã rút {amount}. Số dư còn lại: {\_balance}");

            }

        }

    }

class Program

{

static void Main(string[] args)

{

// TODO: Tạo tài khoản và thử các thao tác

BankAccount acc = new BankAccount();

            acc.Deposit(1000);     // gửi 1000

            acc.Withdraw(300);     // rút 300

            acc.Withdraw(800);     // thử rút quá số dư

            acc.Deposit(-50);      // thử gửi không hợp lệ

            Console.WriteLine($"Số dư cuối cùng: {acc.Balance}");

}

}

}

**TODO 5.4: [KẾ THỪA - ANIMAL & DOG]**

using System;

namespace Inheritance

{

// TODO: Tạo lớp Animal (cha)

// Thuộc tính: Name

// Phương thức: MakeSound() (in ra "Animal makes a sound")

class Animal

    {

        public string Name { get; set; }

        public virtual void MakeSound()

        {

            Console.WriteLine("Animal makes a sound");

        }

    }

// TODO: Tạo lớp Dog (con) kế thừa từ Animal

// Override phương thức MakeSound() (in ra "Woof! Woof!")

// class Dog : Animal

    {

        public override void MakeSound()

        {

            Console.WriteLine("Woof! Woof!");

        }

    }

class Program

{

static void Main(string[] args)

{

// TODO: Tạo đối tượng Animal và Dog

Animal a = new Animal();

            a.Name = "Generic Animal";

            Dog d = new Dog();

            d.Name = "Buddy";

// TODO: Gọi MakeSound() của cả hai

a.MakeSound();

            d.MakeSound();

}

}

}

**📸 CHỨNG THỰC [5.1 -> 5.4]:**

TODO 5.1:

A screenshot of a computer

Description automatically generated

TODO 5.2:

A screenshot of a computer

Description automatically generated

TODO 5.3:

A screenshot of a computer

Description automatically generated

TODO 5.4:

A screenshot of a computer

Description automatically generated

### 🤔 CÂU HỎI PHẢN BIỆN

1. **Tại sao encapsulation (che giấu dữ liệu) lại quan trọng?**

 Bảo vệ dữ liệu khỏi truy cập trái phép.

 Đảm bảo dữ liệu luôn hợp lệ.

 Ẩn sự phức tạp, dễ bảo trì và mở rộng.

1. **Kế thừa vs. Composition - khi nào dùng cái nào?**

 **Kế thừa (is-a):** Dùng khi class con thực sự là class cha (ví dụ Dog is an Animal).

 **Composition (has-a):** Dùng khi class chứa thành phần khác (ví dụ Car has an Engine), linh hoạt và dễ thay đổi.

1. **Lợi ích của polymorphism (đa hình) là gì?**

 Cho phép dùng cùng phương thức nhưng hành vi khác nhau theo đối tượng.

 Mở rộng dễ dàng, giảm sửa code cũ.

 Tái sử dụng code và đơn giản hóa chương trình.

### 💫 KẾT NỐI ĐÁNH GIÁ

**Trọng yếu:** Phần này là **nền tảng OOP** - phải hiểu chắc để học tiếp