Sinh viên: Đỗ Nam Khánh

Lớp: CNTT 18-09

## 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 (Class)** là **khuôn mẫu** (blueprint) định nghĩa các thuộc tính và hành vi chung của một thực thể (tồn tại ở cấp độ logic).

 **Đối tượng (Object)** là **thực thể cụ thể** được tạo ra từ lớp đó và tồn tại trong bộ nhớ với các giá trị riêng biệt.

1. Constructor là gì? Tại sao cần constructor?’

- Là phương thức đặc biệt, **tên trùng với tên lớp**, được gọi **tự động** khi đối tượng được tạo bằng từ khóa new.

- Nó có nhiệm vụ **khởi tạo trạng thái ban đầu** của đối tượng (ví dụ: đặt số dư tài khoản về 0) và bắt buộc người dùng cung cấp các giá trị cần thiết.

1. private, public, protected khác gì?
   * **public** (Công khai): Truy cập từ **bất cứ đâu**.
   * **private** (Riêng tư): Chỉ truy cập được bên trong **chính lớp đó**.
   * **protected** (Được bảo vệ): Truy cập được bên trong **lớp đó** VÀ **các lớp con** (kế thừa).
2. Kế thừa có tác dụng gì?

### 📝 VÍ DỤ THỰC HÀNH

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

using System;

namespace BasicClass

{

    // Tạo lớp Student

    class Student

    {

        // Thuộc tính

        public int StudentId { get; set; }

        // Khắc phục cảnh báo: Khởi tạo Name bằng chuỗi rỗng

        public string Name { get; set; } = string.Empty;

        public double GPA { get; set; }

        // Phương thức in thông tin

        public void Display()

        {

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

        }

    }

    class Program

    {

        static void Main(string[] args)

        {

            // 1. Tạo 2 đối tượng Student và gán giá trị thuộc tính

            //    Sử dụng cú pháp Object Initializer để tối ưu hóa việc gán giá trị

            Console.WriteLine("--- Thông tin Sinh viên ---");

            Student s1 = new Student

            {

                StudentId = 1,

                Name = "Nguyen Van A",

                GPA = 3.2

            };

            Student s2 = new Student

            {

                StudentId = 2,

                Name = "Le Thi B",

                GPA = 3.8

            };

            // 2. Gọi phương thức Display()

            Console.Write("Sinh viên 1: ");

            s1.Display();

            Console.Write("Sinh viên 2: ");

            s2.Display();

            Console.WriteLine("\nHoàn thành chương trình. Nhấn Enter để thoát.");

            Console.ReadLine();

        }

    }

}

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

using System;

namespace Constructor

{

    // Tạo lớp Product với Constructor

    class Product

    {

        // Sử dụng Properties ({ get; set; }) thay vì Field công khai

        public int ProductId { get; set; }

        public string ProductName { get; set; }

        public double Price { get; set; }

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

        {

            this.ProductId = productId;

            this.ProductName = productName;

            this.Price = price;

        }

        // Phương thức Display()

        public void Display()

        {

            Console.WriteLine($"ID: {ProductId}, Name: {ProductName}, Price: {Price:C2}"); // Định dạng Price thành tiền tệ

        }

    }

    class Program

    {

        static void Main(string[] args)

        {

            Console.WriteLine("--- Khởi tạo đối tượng bằng Constructor ---");

            Product p1 = new Product(1, "Laptop Dell XPS", 1850.5);

            Product p2 = new Product(2, "Màn hình Samsung 27\"", 350.99);

            // In thông tin sản phẩm

            Console.Write("Sản phẩm 1: ");

            p1.Display();

            Console.Write("Sản phẩm 2: ");

            p2.Display();

            Console.ReadLine();

        }

    }

}

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

using System;

namespace Encapsulation

{

    // Tạo lớp BankAccount

    class BankAccount

    {

        // Private field: Bảo vệ dữ liệu, chỉ có thể thay đổi qua các phương thức công khai

        private double \_balance;

        // Constructor: Khởi tạo tài khoản với số dư ban đầu là 0

        public BankAccount()

        {

            \_balance = 0;

        }

        // Public property (chỉ đọc): Cho phép truy cập số dư, nhưng không cho phép gán trực tiếp

        public double Balance

        {

            get { return \_balance; }

        }

        // Gửi tiền: Có kiểm tra tính hợp lệ của số tiền

        public void Deposit(double amount)

        {

            if (amount > 0)

            {

                \_balance += amount;

                Console.WriteLine($"Gửi thành công {amount:C2}.");

            }

            else

            {

                Console.WriteLine("Lỗi: Số tiền gửi không hợp lệ (phải > 0).");

            }

        }

        // Rút tiền: Có kiểm tra điều kiện (số tiền hợp lệ và đủ số dư)

        public void Withdraw(double amount)

        {

            if (amount <= 0)

            {

                Console.WriteLine("Lỗi: Số tiền rút không hợp lệ (phải > 0).");

            }

            else if (amount > \_balance)

            {

                Console.WriteLine($"Cảnh báo: Không đủ tiền để rút {amount:C2}. Số dư hiện tại là {\_balance:C2}.");

            }

            else

            {

                \_balance -= amount;

                Console.WriteLine($"Rút tiền thành công {amount:C2}.");

            }

        }

    }

    class Program

    {

        static void Main(string[] args)

        {

            Console.WriteLine("--- Thử nghiệm tính Đóng gói (Encapsulation) ---");

            // Tạo tài khoản

            BankAccount acc = new BankAccount();

            Console.WriteLine($"1. Số dư ban đầu: {acc.Balance:C2}");

            // Gửi tiền

            acc.Deposit(1000);

            Console.WriteLine($"2. Số dư sau khi gửi 1000: {acc.Balance:C2}");

            // Rút tiền hợp lệ

            acc.Withdraw(300);

            Console.WriteLine($"3. Số dư sau khi rút 300: {acc.Balance:C2}");

            // Thử rút vượt số dư

            acc.Withdraw(2000);

            Console.WriteLine($"4. Số dư sau khi thử rút 2000: {acc.Balance:C2}");

            // Thử gửi tiền không hợp lệ

            acc.Deposit(-50);

            Console.ReadLine();

        }

    }

}

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

using System;

namespace Inheritance

{

    // Lớp Animal (cha)

    class Animal

    {

        // Sử dụng Property và khắc phục cảnh báo NRT

        public string Name { get; set; } = string.Empty;

        // Constructor rỗng để khởi tạo Name

        public Animal() { }

        // Constructor có tham số

        public Animal(string name)

        {

            this.Name = name;

        }

        // Đánh dấu phương thức là virtual để cho phép các lớp con ghi đè

        public virtual void MakeSound()

        {

            Console.WriteLine($"{Name} makes a sound");

        }

    }

    // Lớp Dog (con) kế thừa từ Animal

    class Dog : Animal

    {

        // Constructor lớp con, gọi constructor của lớp cha

        public Dog(string name) : base(name)

        {

        }

        // Sử dụng override để ghi đè (thay thế) phương thức của lớp cha

        public override void MakeSound()

        {

            Console.WriteLine($"{Name}: Woof! Woof!");

        }

    }

    // Tạo thêm lớp Cat để thể hiện Đa hình

    class Cat : Animal

    {

        public Cat(string name) : base(name)

        {

        }

        public override void MakeSound()

        {

            Console.WriteLine($"{Name}: Meow! Meow!");

        }

    }

    class Program

    {

        static void Main(string[] args)

        {

            Console.WriteLine("--- Thử nghiệm Kế thừa và Đa hình ---");

            // 1. Tạo đối tượng Animal (Lớp cha)

            Animal genericAnimal = new Animal("Generic Animal");

            genericAnimal.MakeSound(); // Output: Generic Animal makes a sound

            Console.WriteLine("--------------------------");

            // 2. Tạo đối tượng Dog và Cat (Lớp con)

            Dog myDog = new Dog("Bingo");

            myDog.MakeSound(); // Output: Bingo: Woof! Woof!

            Cat myCat = new Cat("Miu");

            myCat.MakeSound(); // Output: Miu: Meow! Meow!

            Console.WriteLine("--------------------------");

            // 3. Minh họa Đa hình (Polymorphism)

            // Tạo một mảng các đối tượng Animal, nhưng lưu trữ Dog và Cat

            Animal[] pets = new Animal[] { myDog, myCat, genericAnimal };

            Console.WriteLine("Kiểm tra Đa hình (Gọi cùng một phương thức trên các đối tượng khác nhau):");

            // Lặp qua mảng, mỗi đối tượng sẽ gọi phiên bản MakeSound() của chính nó

            foreach (Animal pet in pets)

            {

                pet.MakeSound();

            }

            Console.ReadLine();

        }

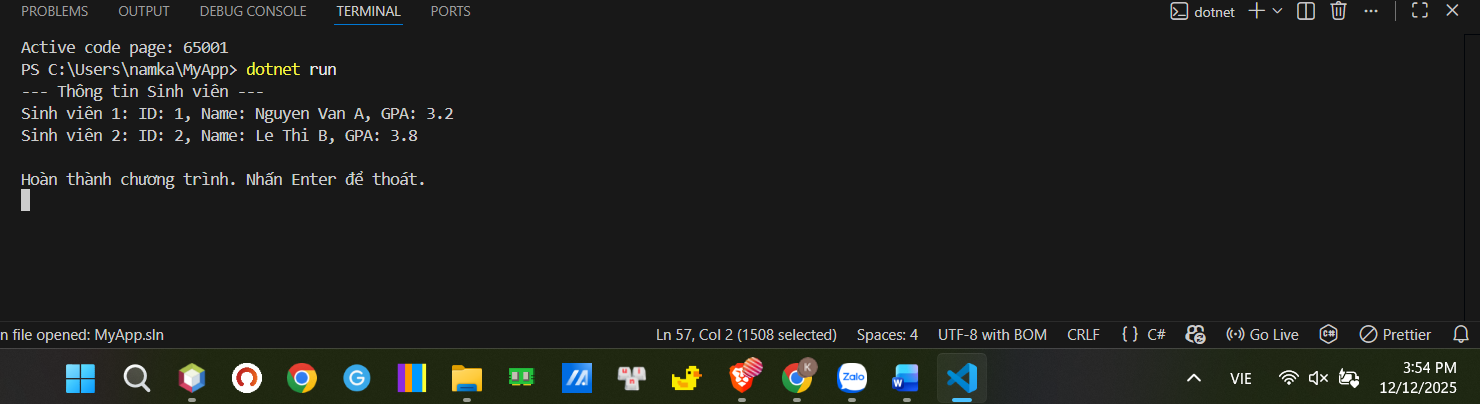
    }

}

**📸 CHỨNG THỰC [5.1 -> 5.4]:**

* Chạy từng chương trình, screenshot output
* Thay đổi dữ liệu để test chức năng

5.1

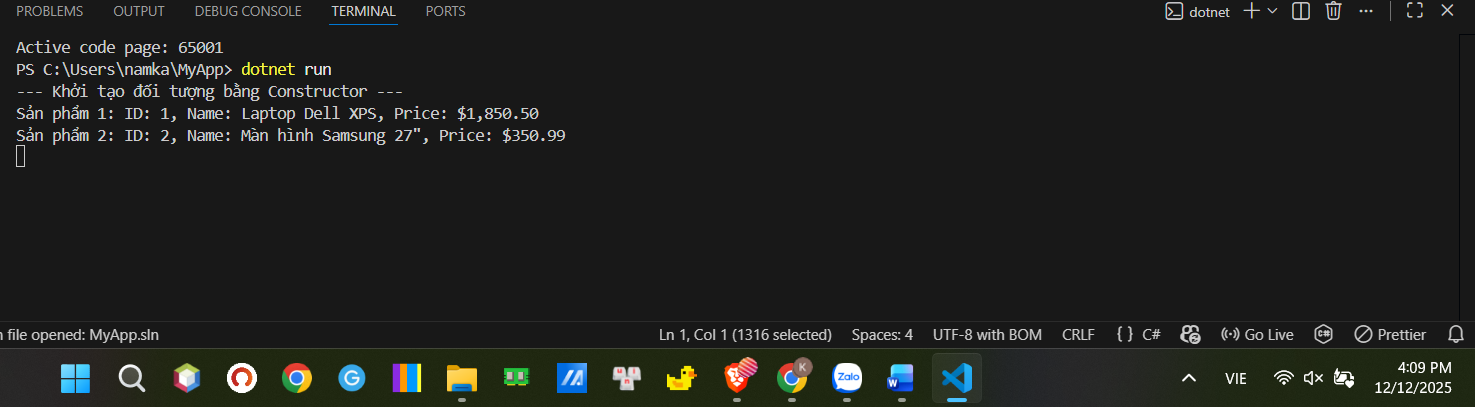


Thay đổi dữ liệu test thức năng

Ảnh có chứa văn bản, phần mềm, Phần mềm đa phương tiện, Phần mềm đồ họa

Nội dung do AI tạo ra có thể không chính xác.

5.2



Thay đổi dữ liệu test thức năng

Ảnh có chứa văn bản, phần mềm, Phần mềm đa phương tiện, Biểu tượng máy tính

Nội dung do AI tạo ra có thể không chính xác.

5.3

Ảnh có chứa văn bản, phần mềm, Phần mềm đa phương tiện, Hệ điều hành

Nội dung do AI tạo ra có thể không chính xác.

Thay đổi dữ liệu để test thức năng

Ảnh có chứa văn bản, phần mềm, Phần mềm đa phương tiện, Phần mềm đồ họa

Nội dung do AI tạo ra có thể không chính xác.

5.4

Ảnh có chứa văn bản, phần mềm, Phần mềm đa phương tiện, Phần mềm đồ họa

Nội dung do AI tạo ra có thể không chính xác.

Thay đổi dữ liệu để test thức năng

Ảnh có chứa văn bản, phần mềm, Phần mềm đa phương tiện, Phần mềm đồ họa

Nội dung do AI tạo ra có thể không chính xác.

### 🤔 CÂU HỎI PHẢN BIỆN

1. **Tại sao encapsulation (che giấu dữ liệu) lại quan trọng?**
   * Đóng gói là cơ chế biến các dữ liệu và hành vi thành một gói duy nhất, đồng thời **bảo vệ** dữ liệu khỏi sự truy cập và thay đổi trái phép.
2. **Kế thừa vs. Composition - khi nào dùng cái nào?**
   * Chỉ dùng Kế thừa khi chắc chắn về mối quan hệ "is-a" và cần Đa hình; còn lại, luôn ưu tiên **Thành phần** để có tính linh hoạt và khả năng bảo trì cao hơn.
3. **Lợi ích của polymorphism (đa hình) là gì?**
   * Lợi ích cốt lõi của Đa hình là khả năng xử lý các đối tượng khác nhau như thể chúng là cùng một loại, tạo ra một kiến trúc phần mềm **linh hoạt, dễ mở rộng và dễ bảo trì**.

### 💫 KẾT NỐI ĐÁNH GIÁ

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