## PHT C# [03] - ĐIỀU KIỆN VÀ VÒNG LẶP

### 🎯 KIẾN THỨC CỐT LỐI

| **Cấu Trúc** | **Cú Pháp** | **Khi Dùng** |
| --- | --- | --- |
| **if** | if (condition) { } | Kiểm tra điều kiện đúng/sai |
| **if-else** | if (c1) { } else { } | Có 2 lựa chọn |
| **if-else if-else** | if (c1) { } else if (c2) { } else { } | Nhiều lựa chọn |
| **switch** | switch(x) { case 1: break; } | Kiểm tra giá trị cụ thể |
| **for** | for(i=0; i<n; i++) { } | Lặp với số lần xác định |
| **while** | while(condition) { } | Lặp cho đến khi điều kiện sai |
| **do-while** | do { } while(condition); | Lặp ít nhất 1 lần |
| **foreach** | foreach(var item in array) { } | Lặp qua mảng |
| **break** | Thoát vòng lặp/switch ngay | Dừng sớm |
| **continue** | Bỏ qua phần còn lại, lần lặp kế tiếp | Bỏ qua những trường hợp nhất định |

### 💡 KHÁM PHÁ KIẾN THỨC

**Câu hỏi tự kiểm tra:**

1. Toán tử so sánh là gì? (==, !=, <, >, <=, >=)
2. Toán tử logic là gì? (&&, ||, !)
3. Sự khác biệt giữa for và foreach là gì?
4. Khi nào dùng break? Khi nào dùng continue?

### 📝 VÍ DỤ THỰC HÀNH

**TODO 3.1: [PHÂN LOẠI ĐIỂM - IF/ELSE IF/ELSE]**

using System;

namespace GradeClassification

{

class Program

{

static void Main(string[] args)

{

// TODO: Nhập điểm (giả sử: 75)

int score = 75; // Thay đổi giá trị để test khác

// TODO: Phân loại điểm

// 90-100: A (Xuất sắc)

// 80-89: B (Khá)

// 70-79: C (Trung bình)

// 60-69: D (Yếu)

// < 60: F (Không đạt)

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// TODO: In kết quả

}

}

}

**TODO 3.2: [SWITCH - CHỌN NGÀY TRONG TUẦN]**

using System;

namespace DayOfWeek

{

class Program

{

static void Main(string[] args)

{

// TODO: Nhập số ngày (1-7)

int day = 3; // Thứ Tư

// TODO: Dùng switch để in tên ngày

// 1: Thứ Hai

// 2: Thứ Ba

// ...

// 7: Chủ Nhật

// default: Ngày không hợp lệ

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

}

}

**TODO 3.3: [VÒNG LẶP FOR - TÍNH TỔNG]**

using System;

namespace SumCalculation

{

class Program

{

static void Main(string[] args)

{

// TODO: Tính tổng các số từ 1 đến 100

// Gợi ý: Dùng vòng lặp for

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// TODO: In kết quả

// "Tổng các số từ 1 đến 100: [kết quả]"

}

}

}

**TODO 3.4: [VÒNG LẶP WHILE - ĐOÁN SỐ]**

using System;

namespace GuessNumber

{

class Program

{

static void Main(string[] args)

{

// TODO: Tạo chương trình đoán số

// - Số bí mật là 50 (cố định)

// - Người chơi đoán (nhập vào 3 lần: 40, 60, 50)

// - Mỗi lần đoán, chương trình báo "Quá thấp" hoặc "Quá cao"

// - Khi đoán đúng, báo "Chính xác!" và thoát

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

}

}

**TODO 3.5: [FOREACH - IN DANH SÁCH]**

using System;

namespace ForeachExample

{

class Program

{

static void Main(string[] args)

{

// TODO: Tạo mảng tên các bạn

string[] friends = { "Mai", "Bình", "Chi", "Danh" };

// TODO: In danh sách bạn bè

// Gợi ý: Dùng foreach

// Output:

// "1. Mai"

// "2. Bình"

// "3. Chi"

// "4. Danh"

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

}

}

**TODO 3.6: [BREAK & CONTINUE - TÌM SỐ]**

using System;

namespace BreakContinue

{

class Program

{

static void Main(string[] args)

{

// TODO: In các số lẻ từ 1 đến 20 (dùng continue)

// Output: 1 3 5 7 9 11 13 15 17 19

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// TODO: Tìm số 7 trong mảng [2, 5, 7, 1, 9, 7, 3]

// Khi tìm thấy, in "Tìm thấy số 7 tại vị trí [chỉ số]" rồi dừng (break)

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

}

}

**📸 CHỨNG THỰC [3.1 -> 3.6]:**

* Chạy 6 chương trình và chụp screenshot output
* Thay đổi input để test các trường hợp khác nhau

### 🤔 CÂU HỎI PHẢN BIỆN

1. **Tại sao vòng lặp for lại được dùng nhiều hơn while trong lập trình?**
2. **Nếu bạn có vòng lặp for lồng trong for, làm cách nào để thoát cả hai vòng lặp cùng lúc?**
3. **So sánh while và do-while - khi nào nên dùng do-while?**

### 💫 KẾT NỐI ĐÁNH GIÁ

**Mốc kiểm tra quan trọng:** Bạn phải **thành thạo if/else/switch** và **for/while/foreach** để học phần tiếp theo

## 

## PHT C# [04] - PHƯƠNG THỨC (FUNCTION/METHOD)

### 🎯 KIẾN THỨC CỐT LỐI

| **Khái Niệm** | **Ý Nghĩa** | **Ví Dụ** |
| --- | --- | --- |
| **Phương thức** | Khối code có tên, có thể tái sử dụng | static int Add(int a, int b) |
| **Tham số (Parameters)** | Biến đầu vào của phương thức | Add(int a, int b) → a, b là tham số |
| **Đối số (Arguments)** | Giá trị cụ thể truyền vào | Add(5, 3) → 5, 3 là đối số |
| **Kiểu trả về** | Kiểu dữ liệu phương thức trả về | int, string, void (không trả về) |
| **Return** | Câu lệnh trả về giá trị | return sum; |
| **Static** | Phương thức thuộc lớp, gọi trực tiếp | Program.Add(5, 3) |
| **Nạp chồng (Overloading)** | Nhiều phương thức cùng tên, tham số khác | Add(int, int) và Add(double, double) |
| **Đệ quy (Recursion)** | Phương thức gọi chính nó | Factorial(n) = n \* Factorial(n-1) |

### 💡 KHÁM PHÁ KIẾN THỨC

**Câu hỏi tự kiểm tra:**

1. Tại sao phải dùng phương thức? (Lợi ích của phương thức là gì?)
2. Sự khác biệt giữa tham số và đối số?
3. Phương thức nào phải có return? Phương thức nào không?
4. Nạp chồng phương thức dựa vào điều gì?

### 📝 VÍ DỤ THỰC HÀNH

**TODO 4.1: [PHƯƠNG THỨC TÍNH TOÁN CỐ BẢN]**

using System;

namespace BasicMethods

{

class Program

{

// TODO: Viết phương thức tính tổng 2 số

// Tên: Add

// Tham số: int a, int b

// Trả về: int

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// TODO: Viết phương thức tính tích 2 số

// Tên: Multiply

// Tham số: double x, double y

// Trả về: double

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

static void Main(string[] args)

{

// TODO: Gọi phương thức Add và in kết quả

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// TODO: Gọi phương thức Multiply và in kết quả

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

}

}

**TODO 4.2: [PHƯƠNG THỨC VOID - KHÔNG TRẢ VỀ]**

using System;

namespace VoidMethods

{

class Program

{

// TODO: Viết phương thức in hộp trang trí

// Tên: PrintBox

// Tham số: string text

// Trả về: void

// Ví dụ: PrintBox("Hello") in ra:

// \*\*\*\*\*\*\*\*\*\*\*

// \* Hello \*

// \*\*\*\*\*\*\*\*\*\*\*

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

static void Main(string[] args)

{

// TODO: Gọi phương thức PrintBox với các giá trị khác nhau

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

}

}

**TODO 4.3: [PHƯƠNG THỨC VỚI MẢNG]**

using System;

namespace ArrayMethods

{

class Program

{

// TODO: Viết phương thức tính tổng các phần tử trong mảng

// Tên: SumArray

// Tham số: int[] numbers

// Trả về: int

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// TODO: Viết phương thức tìm số lớn nhất trong mảng

// Tên: FindMax

// Tham số: int[] numbers

// Trả về: int

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

static void Main(string[] args)

{

int[] scores = { 85, 92, 78, 90, 88 };

// TODO: Gọi SumArray và in kết quả

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// TODO: Gọi FindMax và in kết quả

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

}

}

**TODO 4.4: [NẠP CHỒNG PHƯƠNG THỨC]**

using System;

namespace MethodOverloading

{

class Program

{

// TODO: Viết phương thức Print có 2 phiên bản

// Phiên bản 1: Print(int x) - in một số

// Phiên bản 2: Print(string text) - in một chuỗi

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// TODO: Viết phương thức Add có 2 phiên bản

// Phiên bản 1: Add(int a, int b) - cộng 2 số nguyên

// Phiên bản 2: Add(double a, double b) - cộng 2 số thực

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

static void Main(string[] args)

{

// TODO: Gọi Print với int

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// TODO: Gọi Print với string

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// TODO: Gọi Add(int, int)

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// TODO: Gọi Add(double, double)

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

}

}

**TODO 4.5: [ĐỆ QUY - TÍNH GIAI THỪA]**

using System;

namespace Recursion

{

class Program

{

// TODO: Viết phương thức tính giai thừa

// Tên: Factorial

// Tham số: int n

// Trả về: long

// Công thức: n! = n \* (n-1)!

// Điều kiện dừng: 0! = 1

// Gợi ý: Gọi lại chính nó với n-1

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

static void Main(string[] args)

{

// TODO: Tính 5! bằng Factorial(5) và in kết quả

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// TODO: Tính 10! bằng Factorial(10) và in kết quả

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

}

}

**📸 CHỨNG THỰC [4.1 -> 4.5]:**

* Chạy từng chương trình và chụp output
* Viết comment giải thích logic phương thức

### 🤔 CÂU HỎI PHẢN BIỆN

1. **Tại sao cần nạp chồng phương thức? Không thể dùng tên khác nhau cho từng trường hợp không?**
2. **Đệ quy có ưu và nhược điểm gì?**
   * Khi nào nên dùng đệ quy?
   * Khi nào nên dùng vòng lặp?
3. **Phương thức nên ngắn hay dài? Tại sao?**

### 💫 KẾT NỐI ĐÁNH GIÁ

**Chuẩn bị tiếp theo:** Phiếu 05 sẽ dạy **lập trình hướng đối tượng (OOP)** - Đây sẽ là bước nhảy vọt lớn

## 

## 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?
2. Constructor là gì? Tại sao cần constructor?
3. private, public, protected khác gì?
4. Kế thừa có tác dụng gì?

### 📝 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 Program

{

static void Main(string[] args)

{

// TODO: Tạo 2 đối tượng Student

// TODO: Gán giá trị thuộc tính

// TODO: Gọi phương thức Display()

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

}

}

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

using System;

namespace Constructor

{

// TODO: Tạo lớp Product với Constructor

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

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

class Program

{

static void Main(string[] args)

{

// TODO: Tạo đối tượng Product dùng Constructor

// TODO: In thông tin sản phẩm

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

}

}

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

using System;

namespace Encapsulation

{

// TODO: Tạo lớp BankAccount

// - Private field: \_balance (số dư)

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

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

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

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

class Program

{

static void Main(string[] args)

{

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

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

}

}

**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")

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

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

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

class Program

{

static void Main(string[] args)

{

// TODO: Tạo đối tượng Animal và Dog

// TODO: Gọi MakeSound() của cả hai

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

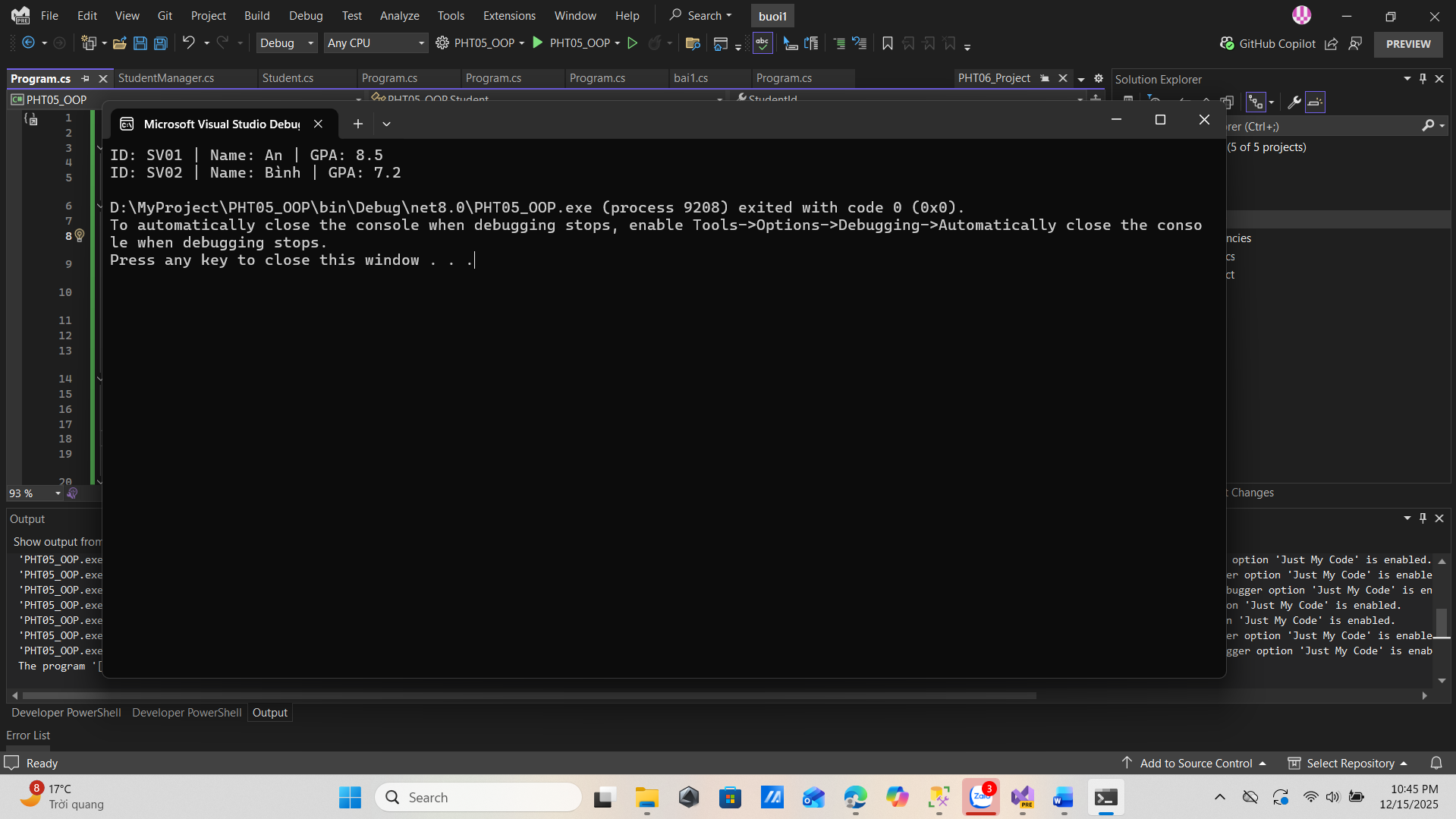
}

}

}

**📸 CHỨNG THỰC [5.1 -> 5.4]:**

TODO 5.1 – LỚP CƠ BẢN



TODO 5.2 – CONSTRUCTOR

A screenshot of a computer

AI-generated content may be incorrect.

TODO 5.3 – ENCAPSULATION

A screenshot of a computer

AI-generated content may be incorrect.

TODO 5.4 – KẾ THỪA

A screenshot of a computer

AI-generated content may be incorrect.

### 🤔 CÂU HỎI PHẢN BIỆN

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

+ Bảo vệ dữ liệu

+ Tránh sửa sai dữ liệu

+ Kiểm soát truy cập

+ Dễ bảo trì hệ thống

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

|  |  |
| --- | --- |
| Kế thừa | Composition |
| Quan hệ is-a | Quan hệ has-a |
| Tái sử dụng code | Linh hoạt hơn |
| Phù hợp lớp cha – con | Phù hợp ghép chức năng |

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

Viết code linh hoạt

+ Dễ mở rộng

+ Giảm phụ thuộc

+ Xử lý nhiều đối tượng theo cùng các

### 💫 KẾT NỐI ĐÁNH GIÁ

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

## PHT C# [06] - XỬ LÝ LỖI VÀ DỰ ÁN TỔNG HỢP

### 🎯 KIẾN THỨC CỐT LỐI

| **Khái Niệm** | **Ý Nghĩa** | **Ví Dụ** |
| --- | --- | --- |
| **Exception** | Lỗi xảy ra lúc chạy chương trình | DivideByZeroException |
| **try-catch** | Bắt lỗi và xử lý | try { } catch { } |
| **finally** | Code luôn chạy, dù có lỗi hay không | Đóng file, giải phóng tài nguyên |
| **Validation** | Kiểm tra dữ liệu đầu vào | Kiểm tra age > 0 |
| **throw** | Ném ra một exception | throw new Exception("Lỗi"); |

### 📝 DỰ ÁN TỔNG HỢP: HỆ THỐNG QUẢN LÝ SINH VIÊN

**Yêu cầu chức năng:**

✅ 1. Danh sách sinh viên (tối đa 50)

✅ 2. Thêm sinh viên mới (có validation)

✅ 3. Xóa sinh viên theo ID

✅ 4. Cập nhật điểm cho sinh viên

✅ 5. Tính điểm trung bình

✅ 6. Tìm điểm cao nhất / thấp nhất

✅ 7. Tìm sinh viên theo ID

✅ 8. Xử lý lỗi (try-catch)

**TODO 6.1: [TẠO LỚP STUDENT]**

// Lớp Student chứa thông tin và điểm của 1 sinh viên

public class Student

{

public string StudentId { get; set; }

public string Name { get; set; }

public double Score { get; set; }

// Constructor

public Student(string id, string name, double score)

{

// TODO: Khởi tạo các thuộc tính với validation

// - StudentId không được rỗng

// - Name không được rỗng

// - Score phải từ 0 đến 10

}

// Phương thức in thông tin

public void Display()

{

// TODO: In ra "ID: [StudentId] | Tên: [Name] | Điểm: [Score]"

}

}

**TODO 6.2: [TẠO LỚP STUDENT MANAGER]**

public class StudentManager

{

private Student[] students = new Student[50];

private int count = 0; // Số lượng sinh viên hiện tại

// TODO: Phương thức AddStudent(string id, string name, double score)

// Thêm sinh viên mới, kiểm tra trùng lặp

// TODO: Phương thức RemoveStudent(string id)

// Xóa sinh viên theo ID

// TODO: Phương thức UpdateScore(string id, double newScore)

// Cập nhật điểm

// TODO: Phương thức GetAverageScore()

// Tính điểm trung bình

// TODO: Phương thức GetMaxScore()

// Tìm điểm cao nhất

// TODO: Phương thức FindStudentById(string id)

// Trả về đối tượng Student hoặc null

// TODO: Phương thức DisplayAllStudents()

// In danh sách tất cả sinh viên

}

**TODO 6.3: [MENU CHÍNH - MAIN PROGRAM]**

using System;

namespace StudentManagementSystem

{

class Program

{

static void Main(string[] args)

{

StudentManager manager = new StudentManager();

bool running = true;

while (running)

{

// TODO: In menu

Console.WriteLine("\n========== MENU ==========");

Console.WriteLine("1. Thêm sinh viên");

Console.WriteLine("2. Xóa sinh viên");

Console.WriteLine("3. Cập nhật điểm");

Console.WriteLine("4. In danh sách");

Console.WriteLine("5. Tính điểm trung bình");

Console.WriteLine("6. Tìm điểm cao nhất");

Console.WriteLine("7. Tìm sinh viên");

Console.WriteLine("0. Thoát");

Console.WriteLine("========================");

// TODO: Nhận lựa chọn từ người dùng

// TODO: Dùng switch xử lý từng lựa chọn

// TODO: Thêm try-catch để xử lý lỗi

}

}

}

}

**📸 CHỨNG THỰC DỰ ÁN:**

* Chụp screenshot menu



* Chụp screenshot các chức năng: thêm, xóa, cập nhật, hiển thị

Thêm:

A screenshot of a computer

AI-generated content may be incorrect.

Sửa:

A black screen with a black background

AI-generated content may be incorrect.

Xóa:

A screenshot of a computer

AI-generated content may be incorrect.

Hiển thị danh sách:

A screenshot of a computer

AI-generated content may be incorrect.

* Chụp screenshot khi có lỗi (input sai), kiểm tra xử lý lỗi

A computer screen shot of a black screen

AI-generated content may be incorrect.

Code check trùng id :

public void AddStudent(string id, string name, double score)

{

if (count >= 50)

throw new Exception("Danh sách đã đầy");

if (FindStudentById(id) != null)

throw new Exception("ID sinh viên đã tồn tại");

students[count++] = new Student(id, name, score);

Console.WriteLine("✅ Thêm sinh viên thành công");

}

A screenshot of a computer

AI-generated content may be incorrect.

Code sủ lý nhập điểm sai

public void UpdateScore(string id, double newScore)

{

if (newScore < 0 || newScore > 10)

throw new Exception("Điểm phải từ 0 - 10");

Student s = FindStudentById(id);

if (s == null)

throw new Exception("Không tìm thấy sinh viên");

s.Score = newScore;

Console.WriteLine("✅ Cập nhật điểm thành công");

}

* File source code hoàn chỉnh

Trong githud

### 🤔 CÂU HỎI PHẢN BIỆN

1. **Nếu người dùng nhập dữ liệu sai (ví dụ: điểm là 15), chương trình nên xử lý như thế nào?**

A computer screen shot of a black screen

AI-generated content may be incorrect.khi người dung nhập hệ thống bắt dầu bắt validate nếu dữ liệu trong khoảng hợp lệ sẽ luu không sẽ bị thông báo lỗi dối với trường hợp sai mầ người dung nhập vào nhưu ví dụ trên

1. \*\*Có thể dùng Listthay vì array không? Ưu điểm gì?\*\*

Có và nên dung list vì cách sử dụng dễ dàng dễ thêm sửa xóa ,thao tác dễ dàng kích thước tự dộng tang chứ ko cố dịnh khi khai báo như array code ngắn gọn lên được ứng dụng nhiều

1. **Nếu muốn lưu dữ liệu xuống file sau khi chương trình thoát, làm sao?**

sử dụng File I/O, có thể lưu dạng text hoặc JSON và load lại khi chương trình khởi động.

### 💫 KẾT NỐI ĐÁNH GIÁ

**Bước tiếp theo:**

* Nếu hoàn thành được dự án này, bạn đã sẵn sàng cho **ASP.NET Core** Backend
* Phiếu tiếp theo sẽ dạy **advanced OOP** (interfaces, abstract class, LINQ)
* Tiếp theo là **xây dựng Web API** thực tế

## 🎓 HƯỚNG DẪN NỘP BÀI

### Cấu Trúc Thư Mục Nộp Bài:

StudentName\_CSHarp/

├── PHT01\_Introduction/

│ ├── HelloWorld.cs

│ └── HelloWorld\_Output.png

├── PHT02\_Variables/

│ ├── ShoppingCalculator.cs

│ ├── StudentGrades.cs

│ ├── UserProfile.cs

│ └── [screenshots]

├── PHT03\_Conditions/

│ ├── GradeClassification.cs

│ ├── DayOfWeek.cs

│ └── [screenshots]

├── PHT04\_Methods/

│ ├── BasicMethods.cs

│ └── [screenshots]

├── PHT05\_OOP/

│ ├── BasicClass.cs

│ └── [screenshots]

└── PHT06\_Project/

├── Student.cs

├── StudentManager.cs

├── Program.cs

└── [screenshots]

### Yêu Cầu Nộp Bài:

* ✅ **Hoàn thành tất cả TODO** trong mỗi phiếu
* ✅ **Screenshot chạy chương trình** cho mỗi bài
* ✅ **Comment code** rõ ràng
* ✅ **Nộp file .cs** + file .md (tài liệu)
* ✅ **Dự án 06** phải **chạy trơn tru**, **xử lý lỗi tốt**

### Tiêu Chí Đánh Giá:

| **Tiêu Chí** | **Điểm** |
| --- | --- |
| Hoàn thành 100% TODO | 40% |
| Code chạy đúng, không lỗi | 30% |
| Xử lý lỗi (try-catch, validation) | 15% |
| Comment, giải thích logic | 10% |
| Nộp đầy đủ file, cấu trúc rõ ràng | 5% |