## 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

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

        {

            // Validation: StudentId không được rỗng

            if (string.IsNullOrWhiteSpace(id))

                throw new ArgumentException("ID không được để trống.");

            // Validation: Name không được rỗng

            if (string.IsNullOrWhiteSpace(name))

                throw new ArgumentException("Tên không được để trống.");

            // Validation: Score phải từ 0 đến 10

            if (score < 0 || score > 10)

                throw new ArgumentException("Điểm phải từ 0 đến 10.");

            StudentId = id;

            Name = name;

            Score = score;

        }

}

// Phương thức in thông tin

public void Display()

{

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

// Phương thức in thông tin

        public void Display()

        {

            Console.WriteLine($"ID: {StudentId} | Tên: {Name} | Điểm: {Score:F2}");

        }

}

}

**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

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

        {

            // Kiểm tra danh sách đầy

            if (count >= students.Length)

                throw new InvalidOperationException("Danh sách sinh viên đã đầy (tối đa 50).");

            // Kiểm tra trùng ID

            if (FindStudentById(id) != null)

                throw new ArgumentException($"ID '{id}' đã tồn tại.");

            // Tạo sinh viên mới (validation trong constructor)

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

            count++;

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

        }

// TODO: Phương thức RemoveStudent(string id)

// Phương thức xóa sinh viên theo ID

        public void RemoveStudent(string id)

        {

            int index = -1;

            for (int i = 0; i < count; i++)

            {

                if (students[i].StudentId == id)

                {

                    index = i;

                    break;

                }

            }

            if (index == -1)

                throw new ArgumentException($"Không tìm thấy sinh viên với ID '{id}'.");

            // Dồn mảng

            for (int i = index; i < count - 1; i++)

            {

                students[i] = students[i + 1];

            }

            students[count - 1] = null;

            count--;

            Console.WriteLine("Xóa sinh viên thành công.");

        }

// Xóa sinh viên theo ID

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

// Cập nhật điểm

// Phương thức cập nhật điểm

        public void UpdateScore(string id, double newScore)

        {

            var student = FindStudentById(id);

            if (student == null)

                throw new ArgumentException($"Không tìm thấy sinh viên với ID '{id}'.");

            if (newScore < 0 || newScore > 10)

                throw new ArgumentException("Điểm phải từ 0 đến 10.");

            student.Score = newScore;

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

        }

// TODO: Phương thức GetAverageScore()

// Tính điểm trung bình

// Phương thức tính điểm trung bình

        public double GetAverageScore()

        {

            if (count == 0)

                throw new InvalidOperationException("Danh sách sinh viên trống.");

            double sum = 0;

            for (int i = 0; i < count; i++)

            {

                sum += students[i].Score;

            }

            return sum / count;

        }

// TODO: Phương thức GetMaxScore()

// Tìm điểm cao nhất

// Phương thức tìm điểm cao nhất

        public double GetMaxScore()

        {

            if (count == 0)

                throw new InvalidOperationException("Danh sách sinh viên trống.");

            double max = students[0].Score;

            for (int i = 1; i < count; i++)

            {

                if (students[i].Score > max)

                    max = students[i].Score;

            }

            return max;

        }

// TODO: Phương thức FindStudentById(string id)

// Trả về đối tượng Student hoặc null

// Phương thức tìm sinh viên theo ID

        public Student FindStudentById(string id)

        {

            for (int i = 0; i < count; i++)

            {

                if (students[i].StudentId == id)

                    return students[i];

            }

            return null;

        }

// TODO: Phương thức DisplayAllStudents()

// In danh sách tất cả sinh viên

// Phương thức in danh sách tất cả sinh viên

        public void DisplayAllStudents()

        {

            if (count == 0)

            {

                Console.WriteLine("Danh sách sinh viên trống.");

                return;

            }

            Console.WriteLine("\n=== DANH SÁCH SINH VIÊN ===");

            for (int i = 0; i < count; i++)

            {

                students[i].Display();

            }

            Console.WriteLine($"Tổng số: {count} 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("========================");

// try

                {

                    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 theo ID");

                    Console.WriteLine("8. Xem số lượng sinh viên");

                    Console.WriteLine("0. Thoát");

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

}

// TODO: Nhận lựa chọn từ người dùng

// Console.Write("Chọn chức năng: ");

                    string choice = Console.ReadLine();

// TODO: Dùng switch xử lý từng lựa chọn

// switch (choice)

                    {

                        case "1": // Thêm sinh viên

                            Console.Write("Nhập ID: ");

                            string id = Console.ReadLine();

                            Console.Write("Nhập tên: ");

                            string name = Console.ReadLine();

                            Console.Write("Nhập điểm (0-10): ");

                            double score = double.Parse(Console.ReadLine());

                            manager.AddStudent(id, name, score);

                            break;

                        case "2": // Xóa sinh viên

                            Console.Write("Nhập ID cần xóa: ");

                            string removeId = Console.ReadLine();

                            manager.RemoveStudent(removeId);

                            break;

                        case "3": // Cập nhật điểm

                            Console.Write("Nhập ID: ");

                            string updateId = Console.ReadLine();

                            Console.Write("Nhập điểm mới (0-10): ");

                            double newScore = double.Parse(Console.ReadLine());

                            manager.UpdateScore(updateId, newScore);

                            break;

                        case "4": // In danh sách

                            manager.DisplayAllStudents();

                            break;

                        case "5": // Tính điểm trung bình

                            double avg = manager.GetAverageScore();

                            Console.WriteLine($"Điểm trung bình của lớp: {avg:F2}");

                            break;

                        case "6": // Tìm điểm cao nhất

                            double max = manager.GetMaxScore();

                            Console.WriteLine($"Điểm cao nhất: {max:F2}");

                            break;

                        case "7": // Tìm sinh viên theo ID

                            Console.Write("Nhập ID cần tìm: ");

                            string findId = Console.ReadLine();

                            var student = manager.FindStudentById(findId);

                            if (student != null)

                            {

                                Console.WriteLine("\nThông tin sinh viên:");

                                student.Display();

                            }

                            else

                                Console.WriteLine("Không tìm thấy sinh viên.");

                            break;

                        case "8": // Xem số lượng sinh viên

                            int currentCount = manager.GetStudentCount();

                            Console.WriteLine($"Số lượng sinh viên hiện tại: {currentCount}/50");

                            break;

                        case "0": // Thoát

                            running = false;

                            Console.WriteLine("\nChương trình kết thúc.");

                            Console.WriteLine("Cảm ơn bạn đã sử dụng!");

                            break;

                        default:

                            Console.WriteLine("Lựa chọn không hợp lệ. Vui lòng chọn từ 0-8.");

                            break;

                    }

// TODO: Thêm try-catch để xử lý lỗi

// catch (FormatException)

                {

                    Console.WriteLine("Lỗi: Dữ liệu nhập không đúng định dạng (ví dụ: điểm phải là số).");

                }

                catch (ArgumentException ex)

                {

                    Console.WriteLine($"Lỗi nhập liệu: {ex.Message}");

                }

                catch (InvalidOperationException ex)

                {

                    Console.WriteLine($"Lỗi hoạt động: {ex.Message}");

                }

                catch (Exception ex)

                {

                    Console.WriteLine($"Lỗi không xác định: {ex.Message}");

                }

}

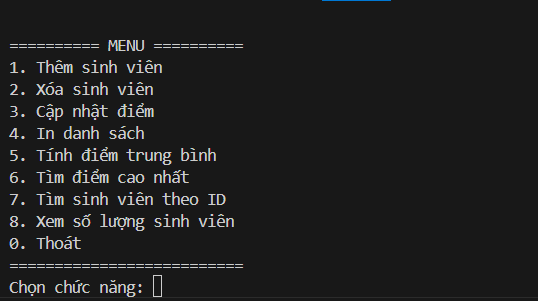
}

}

}

**📸 CHỨNG THỰC DỰ ÁN:**

* Chụp screenshot menu



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

+ thêm:



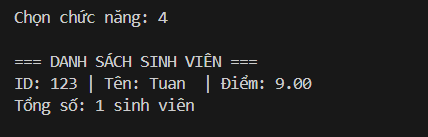
+ xóa:



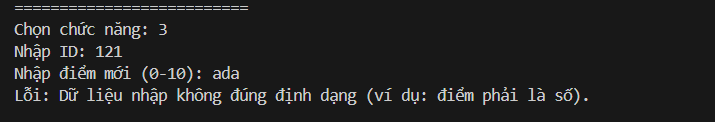
+ cập nhật:



+ hiển thị:



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



* File source code hoàn chỉnh

### 🤔 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?**

Đã xử lý trong constructor của Student và phương thức UpdateScore, ném ArgumentException nếu điểm ngoài [0, 10].

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

Có thể dùng List<Student> để tự động mở rộng, không giới hạn 50 sinh viên, dễ thêm/xóa hơn.

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

Có thể dùng System.IO.File.WriteAllText với định dạng JSON/CSV, hoặc BinaryFormatter để lưu danh sách sinh viên.

### 💫 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

### 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**

### 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]

### 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% |