## 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]**

using System;

namespace StudentManagementSystem

{

*//Todo 6.1 tạo lớp student và kiểm tra điều kiện*

    public class Student

    {

        private string \_StudentsID = string.Empty;

        private string \_Name = string.Empty;

        private double \_Score;

        public string StudentsID

        {

            get {return \_StudentsID;}

            set

            {

                if (String.IsNullOrWhiteSpace(value))

                {

                    throw new ArgumentException("bạn đã để trống");

                }

                if (!value.StartsWith("SV"))

                {

                    throw new ArgumentException("Mã SV phải bắt đầu từ Mx 'SV'");

                }

                string numberpart = value.Substring(2);

                if (!int.TryParse(numberpart, out int id) || id < 1)

                {

                    throw new ArgumentException("Mã sinh viên phải có số >= 01");

                }

                \_StudentsID = value;

            }

        }

        public string Name

        {

            get{return \_Name;}

            set

            {

                if (string.IsNullOrWhiteSpace(value))

                {

                    throw new ArgumentException("bạn không được để trống tên");

                }

                \_Name = value;

            }

        }

        public double Score

        {

            get {return \_Score;}

            set

            {

                if(value < 0 || value > 10)

                {

                    throw new ArgumentException("Bạn nhập điểm thấp hơn 0 hoặc hớn hơn 10");

                }

                \_Score = value;

            }

        }

*// constructor*

        public Student(string *studentsid*, string *name*, double *score*)

        {

            StudentsID = *studentsid*;

            Name = *name*;

            Score = *score*;

        }

*//method*

        public void Display()

        {

            Console.WriteLine($"ID: [{StudentsID}] | Name: [{Name}] | Score: [{Score}]");

        }

    }

}

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

using System;

namespace StudentManagementSystem

{

*//TODO 6.2 tạo lớp StudentManager*

    public class Program

    {

        private string \_StudentsID = string.Empty;

        private string \_Name = string.Empty;

        private double \_Score;

        public string StudentsID

        {

            get {return \_StudentsID;}

            set

            {

                if (String.IsNullOrWhiteSpace(value))

                {

                    throw new ArgumentException("bạn đã để trống");

                }

                if (!value.StartsWith("SV"))

                {

                    throw new ArgumentException("Mã SV phải bắt đầu từ Mx 'SV'");

                }

                string numberpart = value.Substring(2);

                if (!int.TryParse(numberpart, out int id) || id < 1)

                {

                    throw new ArgumentException("Mã sinh viên phải có số >= 01");

                }

                \_StudentsID = value;

            }

        }

        public string Name

        {

            get{return \_Name;}

            set

            {

                if (string.IsNullOrWhiteSpace(value))

                {

                    throw new ArgumentException("bạn không được để trống tên");

                }

                \_Name = value;

            }

        }

        public double Score

        {

            get {return \_Score;}

            set

            {

                if(value < 0 || value > 10)

                {

                    throw new ArgumentException("Bạn nhập điểm thấp hơn 0 hoặc hớn hơn 10");

                }

                \_Score = value;

            }

        }

*// constructor*

        public void Student(string *studentsid*, string *name*, double *score*)

        {

            StudentsID = *studentsid*;

            Name = *name*;

            Score = *score*;

        }

*//method*

        public void Display()

        {

            Console.WriteLine($"ID: [{StudentsID}] | Name: [{Name}] | Score: [{Score}]");

        }

    }

*// Todo 6.2 Tạo lớp quản lý sinh viên*

    public class StudentManager

    {

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

        private int count = 0;

*//method*

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

        {

            if (count >= students.Length)

            {

                Console.WriteLine($"Danh sách sinh viên đã đầy");

                    return;

            }

*// kiểm tra trùng lặp ID*

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

            {

                if (students[i].StudentsID == *id*)

                {

                    Console.WriteLine($"Sinh viên với ID {*id*}");

                    return;

                }

            }

*// thêm sinh viên mới*

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

        count++;

        Console.WriteLine($"Đã thêm sinh viên thành công");

        }

*// hiển thị toàn bộ danh sách*

        public void DisplayAll()

        {

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

            {

                students[i].Display();

            }

        }

*// method xóa sinh viên theo id*

        public void RemoveStudent(string *id*)

        {

            int index = -1;

*// tìm vị trí sinh viên có ID cần xóa*

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

            {

                if (students[i].StudentsID == *id*)

                {

                    index = i;

                    break;

                }

            }

            if (index == -1)

            {

                Console.WriteLine($"không tìm thấy sinh viên với ID {*id*}");

                return;

            }

*// dịch các phần tử phía sau lên*

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

            {

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

            }

*//giảm số lượng*

            count--;

            Console.WriteLine($"Đã xóa sinh viên với ID {*id*}");

        }

        public void UpdateScore(string *id*, double *newscore*)

        {

            bool found = false;

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

            {

                if (students[i].StudentsID == *id*)

                {

                    if (*newscore* < 0 || *newscore* > 10)

                    {

                        Console.WriteLine($"điểm phải nằm trong khu vực từ 0 - 10");

                    }

                    students[i].Score = *newscore*;

                    Console.WriteLine($"đã cập nhập điểm cho sinh viên");

                    found = true;

                    break;

                }

            }

            if (!found)

            {

                Console.WriteLine($"Không tìm thấy sinh viên với mã ID {*id*}");

            }

        }

        public double GetAverageScore()

        {

            if (count == 0)

            {

                Console.WriteLine($"Danh sách sinh viên trống, không có điểm trung bình");

                return 0;

            }

            double sum = 0;

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

            {

                sum += students[i].Score;

            }

            double avg = sum / count;

            return avg;

        }

        public double GetMaxScore()

        {

            if (count == 0)

            {

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

                return 0;

            }

            double maxScore = students[0].Score; *// giả sử phần tử đầu tiên lẫm*

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

            {

                if (students[i].Score > maxScore)

                {

                    maxScore = students[i].Score;

                }

            }

            return maxScore;

        }

        public Student? FindStudentByID(string *id*)

        {

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

            {

                if (students[i]. StudentsID == *id*)

                {

                    return students[i]; *//trả về đối tượng student tìm thấy*

                }

            }

            return null; *// không tìm thấy*

        }

    }

}

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

using System;

namespace StudentManagementSystem

{

    public class StudentManagement

    {

        public static void Main(string[] *args*)

        {

            StudentManager sm1 = 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*

*// đọc phím bấm*

                Console.WriteLine($"Nhập lựa chọn: ");

                string? input = Console.ReadLine();

*// kiểm tra hợp lệ*

                int choice;

                if (!int.TryParse(input, out choice))

                {

                    Console.WriteLine($"Vui lòng nhập số từ 0 đến 7");

                    continue;

                }

                switch (choice)

                {

                    case 1:

                        {

                            sm1.AddStudent("SV01", "A", 9.4); *// thêm sinh viên vào mảng*

                            sm1.AddStudent("SV02", "B", 9.2);

                            sm1.AddStudent("SV03", "C", 8.5);

                            sm1.DisplayAll();

                            break;

                        }

                    case 2:

                        {

                            sm1.RemoveStudent("SV02"); *// xóa sinh viên theo ID*

                            sm1.DisplayAll();

                            break;

                        }

                    case 3:

                        {

                            sm1.UpdateScore("SV01", 9.2); *// cập nhật điểm*

                            sm1.DisplayAll();

                            break;

                        }

                    case 4:

                        {

                            double average = sm1.GetAverageScore(); *// điểm trung bình lớp*

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

                            break;

                        }

                    case 5 :

                        {

                            double max = sm1.GetMaxScore(); *// điểm cao nhất lớp*

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

                            break;

                        }

                    case 6:

                        {

*// tìm sinh viên theo ID*

                            Student? st = sm1.FindStudentByID("SV01");

                            if (st != null)

                            {

                                Console.WriteLine($"Đã tìm thấy sinh viên");

                                st.Display();

                            }

                            else

                            {

                                Console.WriteLine($"Không tìm thấy sinh viên với ID SV01");

                            }

*// thử tìm ID không tồn tại*

                            Student? st2 = sm1.FindStudentByID("SV67");

                            if (st2 == null)

                            {

                                Console.WriteLine($"Không tìm thấy sinh viên với ID SV67");

                            }

                            break;

                        }

                    case 7 :

                        {

*// hiển thị tất cả sinh viên trong danh sách*

                            sm1.DisplayAll();

                            break;

                        }

                    default :

                        {

                            Console.WriteLine($"Đang thoát");

                            running = false;

                            break;

                        }

                }

                break;

            }

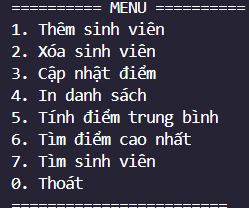
        }

    }

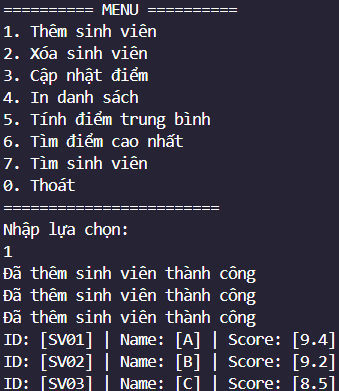
}

**📸 CHỨNG THỰC DỰ ÁN:**

Kết quả Menu



Kết quả chức năng 1

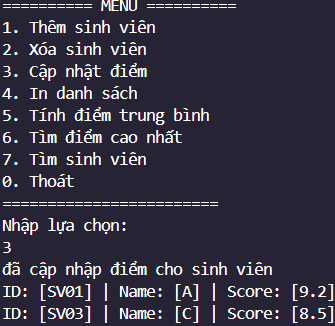


Kết quả chức năng 2

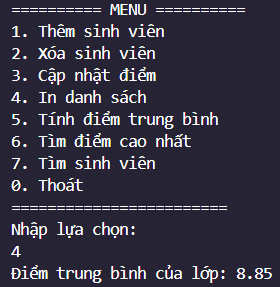
A screenshot of a computer program

AI-generated content may be incorrect.

Kết quả chức năng 3



Kết quả chức năng 4

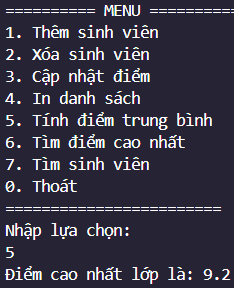


Kết quả chức năng 4

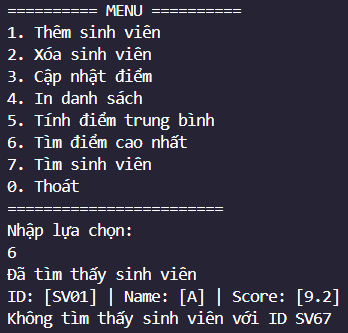
A screenshot of a computer menu

AI-generated content may be incorrect.

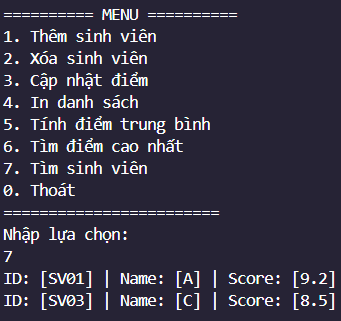
Kết quả chức năng 5



Kết quả chức năng 6



Kết quả chức năng 7



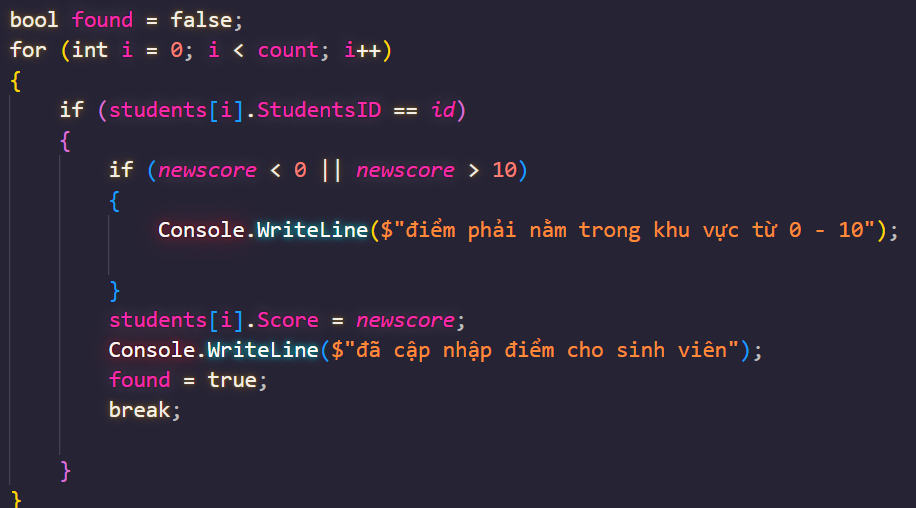
Kết quả khi thoát

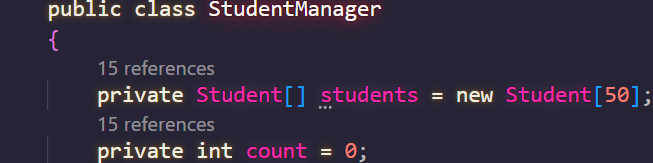
A screenshot of a computer program

AI-generated content may be incorrect.

* Chụp screenshot menu
* Chụp screenshot các chức năng: 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?**
   * Khi nhập dữ liệu sai (nằm ngoài điều kiện của if thì sẽ báo lỗi
   * 
2. \*\*Có thể dùng Listthay vì array không? Ưu điểm gì?\*\*
   * Hoàn toàn có thể List <students> thay vì mảng students[]



* + Ưu điểm của list là kích thước động, thao tác tiện lợi, dễ đọc dễ bảo trì, LINQ hỗ trợ mạnh mẽ

1. **Nếu muốn lưu dữ liệu xuống file sau khi chương trình thoát, làm sao?**
   * Nếu lưu file JSON thì tạo thêm 1 method SaveToFile có tham chiếu là string path. Đặt 1 tên biến tên json = Jsonserializer.Serialize(students); rồi sử dụng lệnh File.WriteAllText(path, json);

### 💫 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