Day 6 – DAY 7: Mini Project & Weekly Recap **Date:** 15-10-2025

Objective: To apply all concepts learned during Week 1 — C#, OOP, Collections, LINQ, and SQL — by developing a small integrated console-based project that demonstrates both programming logic and basic database understanding

1. Weekly Recap – Core Concepts

1.1 C# Fundamentals

- **CLR (Common Language Runtime):** Executes .NET programs and manages memory, exceptions, and security.
- **JIT (Just-In-Time Compiler):** Converts intermediate CIL code into native machine code at runtime for performance.
- **Garbage Collector:** Automatically manages memory, removing unused objects.
- **Managed Code:** Code executed under CLR supervision for safe and efficient memory use.
- **Data Types & Variables:**
  - Value types (int, float, bool) store data directly.
  - Reference types (string, object, class) store object references.
- **Operators:** Arithmetic, relational, logical, assignment, and ternary used for calculations and decisions.
- **Conditional Statements:** if, else if, switch – execute code blocks based on conditions.
- **Loops:** for, while, do-while, and foreach – used for repetitive operations.
- **Methods:** Reusable blocks of code, can have parameters, return types, and optional/out parameters.

1.2 Object-Oriented Programming (OOP)

- **Classes:** Blueprints for creating objects.
- **Objects:** Instances of classes.
- **Constructors:** Special methods used to initialize objects.
- **Encapsulation:** Hiding data inside classes using private members and exposing through properties.
- **Inheritance:** Allows a class to derive from another to reuse and extend functionality.
- **Polymorphism:** Same method behaves differently based on context (method overriding/overloading).

- **Abstraction:** Hiding complex details and exposing only necessary parts (using abstract classes or interfaces).
- **Interfaces:** Define contract-like behavior for classes that implement them.

1.3 Collections & LINQ

- **Arrays:** Fixed-size collection of similar data types.
- **Lists:** Dynamic-size collections (List<T>).
- **Dictionaries:** Key-value pairs for fast lookups (Dictionary<TKey, TValue>).
- **foreach Loop:** Used to iterate over collections.
- **LINQ (Language Integrated Query):** Simplifies data manipulation with methods like Select(), Where(), OrderBy().
- **Lambda Expressions:** Short-hand anonymous functions used in LINQ queries.

Example:

```
var highScores = students.Where(s => s.Marks > 75).Select(s => s.Name);
```

1.4 SQL & RDBMS

- **RDBMS (Relational Database Management System):** Stores data in tables with relationships.
- **Primary Key:** Uniquely identifies each record.
- **Foreign Key:** Creates relation between tables.
- **Normalization:** Organizing data to remove redundancy (1NF, 2NF, 3NF).
- **CRUD Operations:**
  - SELECT → Retrieve data
  - INSERT → Add new data
  - UPDATE → Modify data
  - DELETE → Remove data
- **JOINS:**
  - INNER JOIN – matches data in both tables
  - LEFT JOIN – all from left + matching right
  - RIGHT JOIN – all from right + matching left

- FULL JOIN – all from both sides

Example:

```
SELECT Students.Name, Grades.Subject, Grades.Marks
FROM Students
INNER JOIN Grades ON Students.Id = Grades.StudentId;
```

2. Mini Project Overview – Student Grade Management System

Goal

Create a **C# console application** that:

1. Uses **OOP principles** (classes, inheritance, interfaces)
2. Uses **Collections and LINQ** for data management
3. Simulates a **database-like structure**
4. Performs CRUD operations

2.1 Features

- Add student details (ID, Name, Subject, Marks)
- Display all students
- Filter students based on marks using LINQ
- Calculate average marks
- Identify top-performing student

2.2 Recommended Class Structure

1. **Student.cs**
    - Properties: Id, Name, Subject, Marks
    - Constructor for initialization
2. **IStudentService.cs (Interface)**
    - Methods: AddStudent(), ViewStudents(), GetTopper(), CalculateAverage()
3. **StudentService.cs**
    - Implements interface methods using List<Student>

- LINQ for filtering and sorting

4. **Program.cs**
    - Main menu for user interaction (Add/View/Topper/Exit)

2.3 Example Code Snippets

Student.cs

```
public class Student
{
    public int Id { get; set; }
    public string Name { get; set; }
    public string Subject { get; set; }
    public double Marks { get; set; }

    public Student(int id, string name, string subject, double marks)
    {
        Id = id;
        Name = name;
        Subject = subject;
        Marks = marks;
    }
}
```

IStudentService.cs

```
public interface IStudentService
{
    void AddStudent(Student student);
    void ViewStudents();
    Student GetTopper();
    double CalculateAverage();
}
```

StudentService.cs

```
using System;
using System.Collections.Generic;
using System.Linq;

public class StudentService : IStudentService
{
    private List<Student> students = new List<Student>();

    public void AddStudent(Student student)
    {
        students.Add(student);
        Console.WriteLine("Student added successfully.\n");
    }

    public void ViewStudents()
    {
        foreach (var s in students)
            Console.WriteLine($"ID: {s.Id}, Name: {s.Name}, Subject: {s.Subject}, Marks:
{s.Marks}");
    }

    public Student GetTopper()
    {
        return students.OrderByDescending(s => s.Marks).FirstOrDefault();
    }

    public double CalculateAverage()
    {
        return students.Average(s => s.Marks);
```

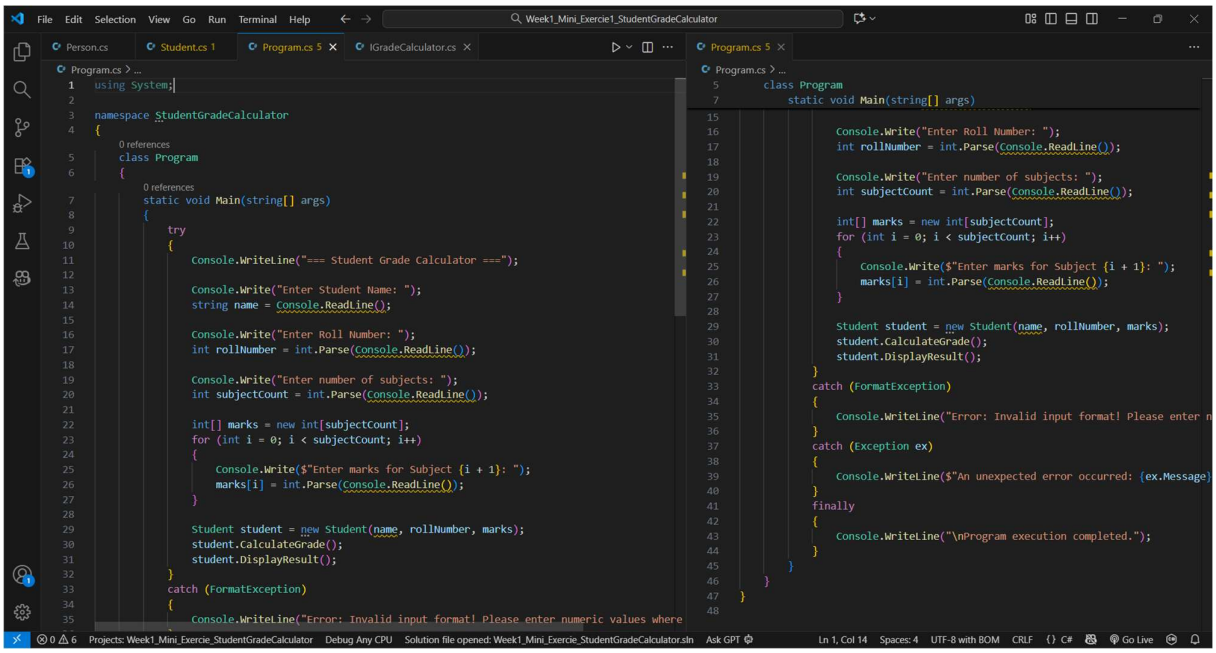
```
    }
}
```

Program.cs

```
class Program
{
    static void Main(string[] args)
    {
        IStudentService service = new StudentService();
        while (true)
        {
            Console.WriteLine("\n--- Student Grade Management System ---");
            Console.WriteLine("1. Add Student");
            Console.WriteLine("2. View All Students");
            Console.WriteLine("3. Show Topper");
            Console.WriteLine("4. Calculate Average");
            Console.WriteLine("5. Exit");
            Console.Write("Enter your choice: ");
            int choice = Convert.ToInt32(Console.ReadLine());

            switch (choice)
            {
                case 1:
                    Console.Write("Enter ID: ");
                    int id = Convert.ToInt32(Console.ReadLine());
                    Console.Write("Enter Name: ");
                    string name = Console.ReadLine();
                    Console.Write("Enter Subject: ");
                    string subject = Console.ReadLine();
                    Console.Write("Enter Marks: ");
```

```
                double marks = Convert.ToDouble(Console.ReadLine());
                service.AddStudent(new Student(id, name, subject, marks));
                break;

            case 2:
                service.ViewStudents();
                break;

            case 3:
                var topper = service.GetTopper();
                Console.WriteLine($"Topper: {topper.Name} ({topper.Marks})");
                break;

            case 4:
                Console.WriteLine($"Average Marks: {service.CalculateAverage()}");
                break;

            case 5:
                return;
        }
    }
  }
}
```

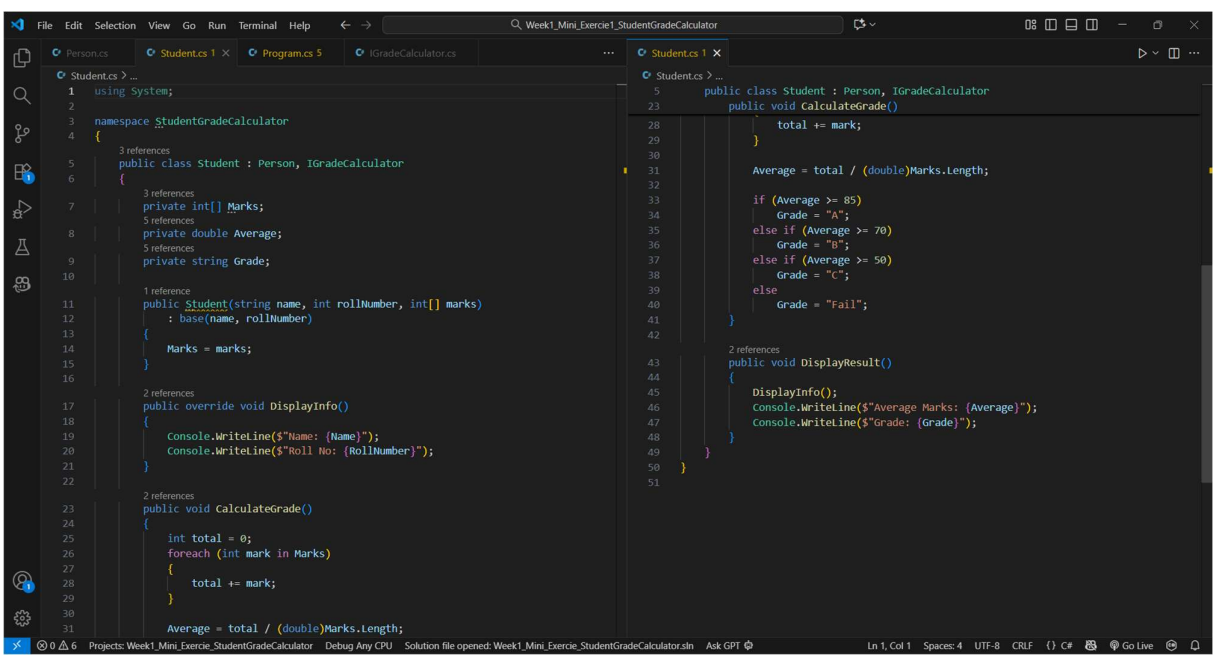
Snapshots:



```
using System;

namespace StudentGradeCalculator
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                Console.WriteLine("=== Student Grade Calculator ===");

                Console.Write("Enter Student Name: ");
                string name = Console.ReadLine();

                Console.Write("Enter Roll Number: ");
                int rollNumber = int.Parse(Console.ReadLine());

                Console.Write("Enter number of subjects: ");
                int subjectCount = int.Parse(Console.ReadLine());

                int[] marks = new int[subjectCount];
                for (int i = 0; i < subjectCount; i++)
                {
                    Console.Write($"Enter marks for Subject {i + 1}: ");
                    marks[i] = int.Parse(Console.ReadLine());
                }

                Student student = new Student(name, rollNumber, marks);
                student.CalculateGrade();
                student.DisplayResult();
            }
            catch (FormatException)
            {
                Console.WriteLine("Error: Invalid input format! Please enter numeric values where
```

```
    class Program
        static void Main(string[] args)

                Console.Write("Enter Roll Number: ");
                int rollNumber = int.Parse(Console.ReadLine());

                Console.Write("Enter number of subjects: ");
                int subjectCount = int.Parse(Console.ReadLine());

                int[] marks = new int[subjectCount];
                for (int i = 0; i < subjectCount; i++)
                {
                    Console.Write($"Enter marks for Subject {i + 1}: ");
                    marks[i] = int.Parse(Console.ReadLine());
                }

                Student student = new Student(name, rollNumber, marks);
                student.CalculateGrade();
                student.DisplayResult();
            }
            catch (FormatException)
            {
                Console.WriteLine("Error: Invalid input format! Please enter n
            }
            catch (Exception ex)
            {
                Console.WriteLine($"An unexpected error occurred: {ex.Message}
            }
            finally
            {
                Console.WriteLine("\nProgram execution completed.");
            }
        }
    }
}
```

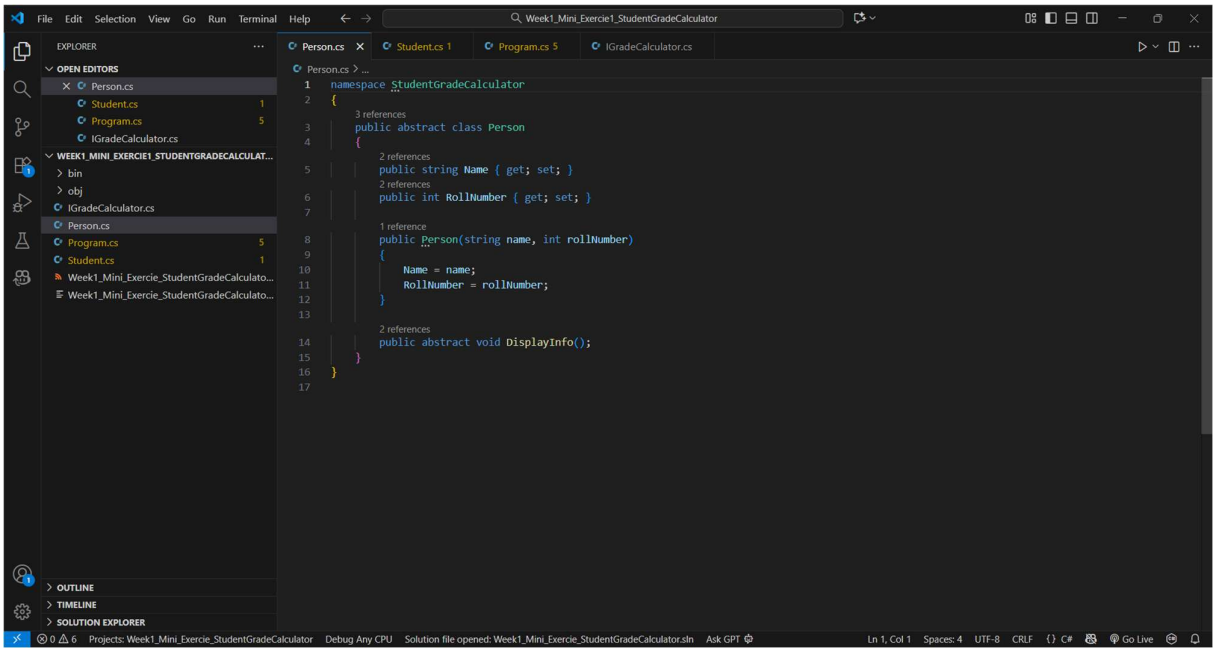
Code : Program.cs



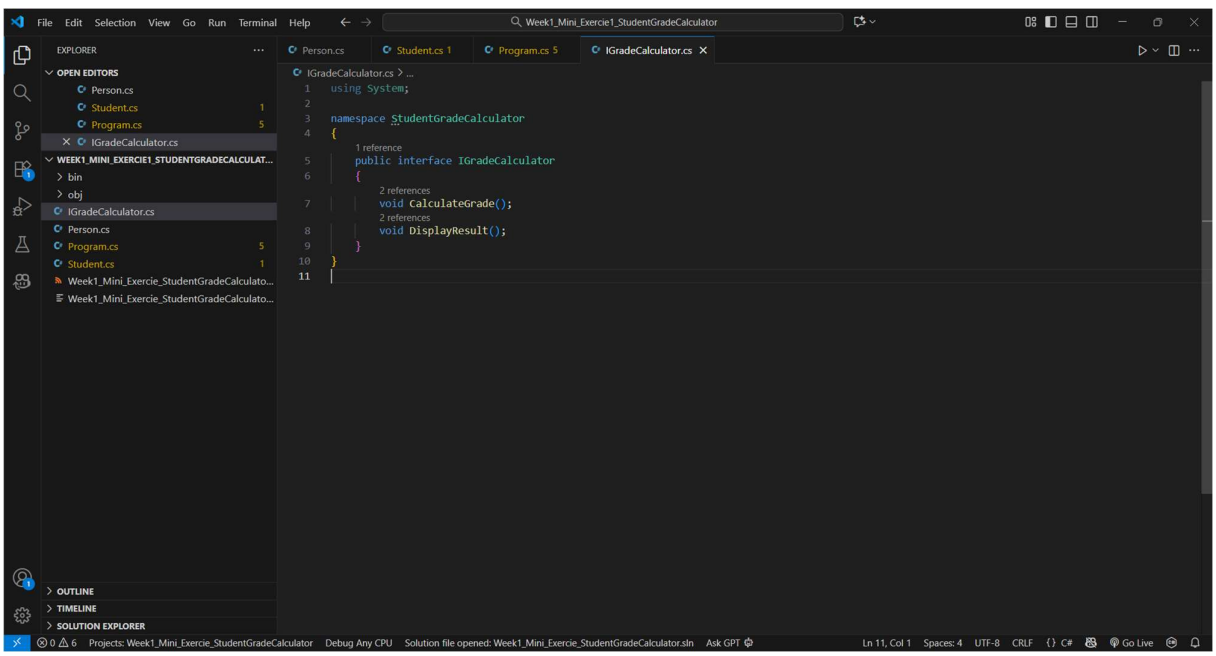
```
using System;

namespace StudentGradeCalculator
{
    public class Student : Person, IGradeCalculator
    {
        private int[] Marks;
        private double Average;
        private string Grade;

        public Student(string name, int rollNumber, int[] marks)
            : base(name, rollNumber)
        {
            Marks = marks;
        }

        public override void DisplayInfo()
        {
            Console.WriteLine($"Name: {Name}");
            Console.WriteLine($"Roll No: {RollNumber}");
        }

        public void CalculateGrade()
        {
            int total = 0;
            foreach (int mark in Marks)
            {
                total += mark;
            }

            Average = total / (double)Marks.Length;
```

```
    public class Student : Person, IGradeCalculator
        public void CalculateGrade()
                total += mark;
            }

            Average = total / (double)Marks.Length;

            if (Average >= 85)
                Grade = "A";
            else if (Average >= 70)
                Grade = "B";
            else if (Average >= 50)
                Grade = "C";
            else
                Grade = "Fail";
        }

        public void DisplayResult()
        {
            DisplayInfo();
            Console.WriteLine($"Average Marks: {Average}");
            Console.WriteLine($"Grade: {Grade}");
        }
    }
}
```

Code : Student.cs



Code : Person.cs



Code : IGradeCalculator.cs

```
=== Student Grade Calculator ===
Enter Student Name: Uday Shetty
Enter Roll Number: 145
Enter number of subjects: 6
Enter marks for Subject 1: 99
Enter marks for Subject 2: 97
Enter marks for Subject 3: 86
Enter marks for Subject 4: 98
Enter marks for Subject 5: 97
Enter marks for Subject 6: 95
Name: Uday Shetty
Roll No: 145
Average Marks: 95.33333333333333
Grade: A
```

Final Output : Avg of Marks