**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;

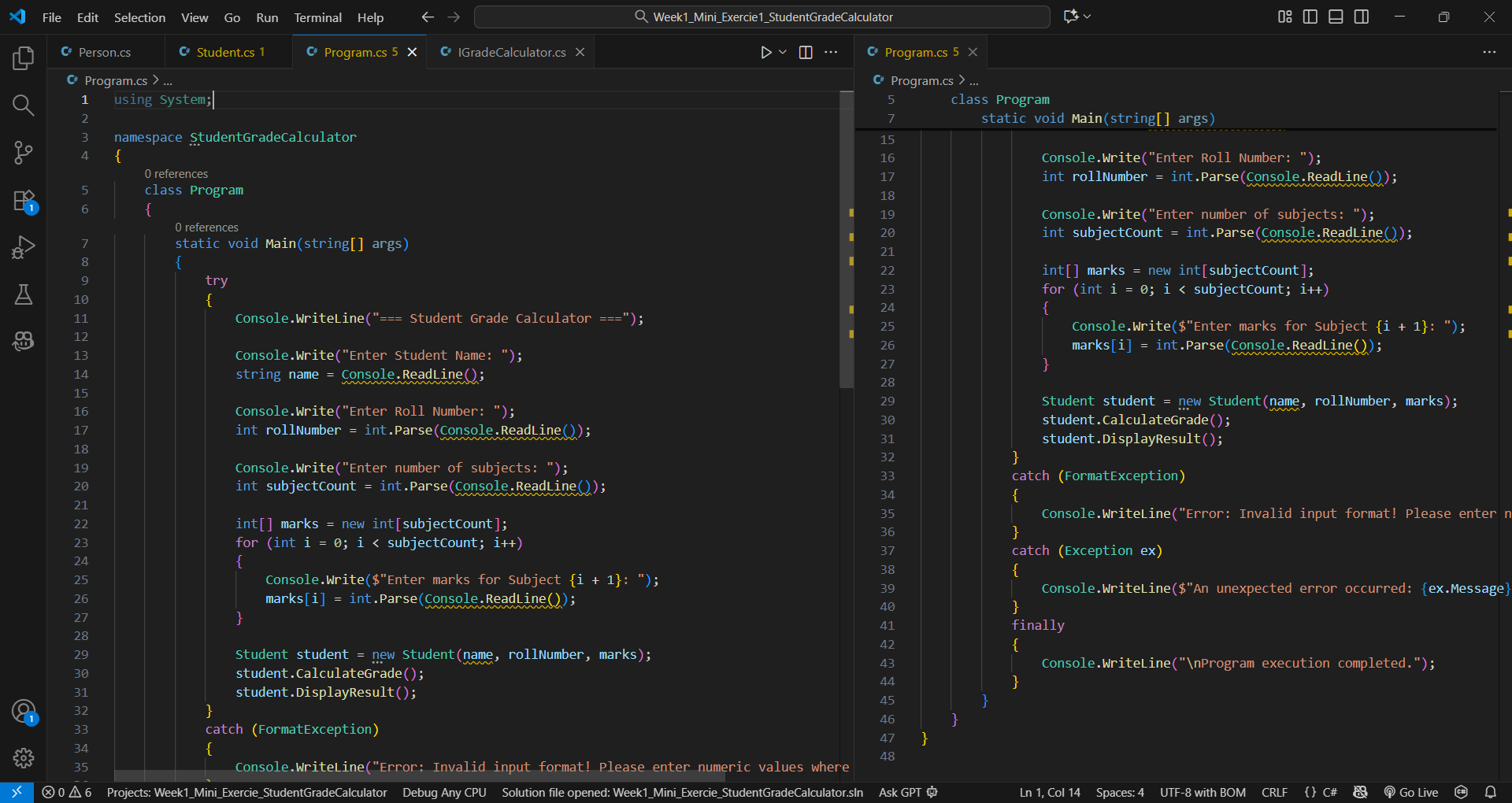
}

}

}

}

**Snapshots:**



Code : Program.cs

A screenshot of a computer

AI-generated content may be incorrect.

Code : Student.cs

A screenshot of a computer

AI-generated content may be incorrect.

Code : Person.cs

A screenshot of a computer

AI-generated content may be incorrect.

Code : IGradeCalculator.cs

A computer screen shot of a number

AI-generated content may be incorrect.

Final Output : Avg of Marks