**Yash Raj**

DN 4.0 Deep Skilling Week 1 Assignment Submission

SuperSet ID - (**6364442**)

Exercise 1: Implementing the Singleton Pattern

Module 1 - Design Patterns and Principles

using System;

public sealed class Singleton

{

    private static readonly Lazy<Singleton> instance = new Lazy<Singleton>(() => new Singleton());

    private Singleton()

    {

        Console.WriteLine("Singleton Instance Created");

    }

    public static Singleton Instance

    {

        get

        {

            return instance.Value;

        }

    }

    public void DoSomething()

    {

        Console.WriteLine("Singleton method called.");

    }

}

class Program

{

    static void Main(string[] args)

    {

        Singleton s1 = Singleton.Instance;

        Singleton s2 = Singleton.Instance;

        s1.DoSomething();

        Console.WriteLine(ReferenceEquals(s1, s2)

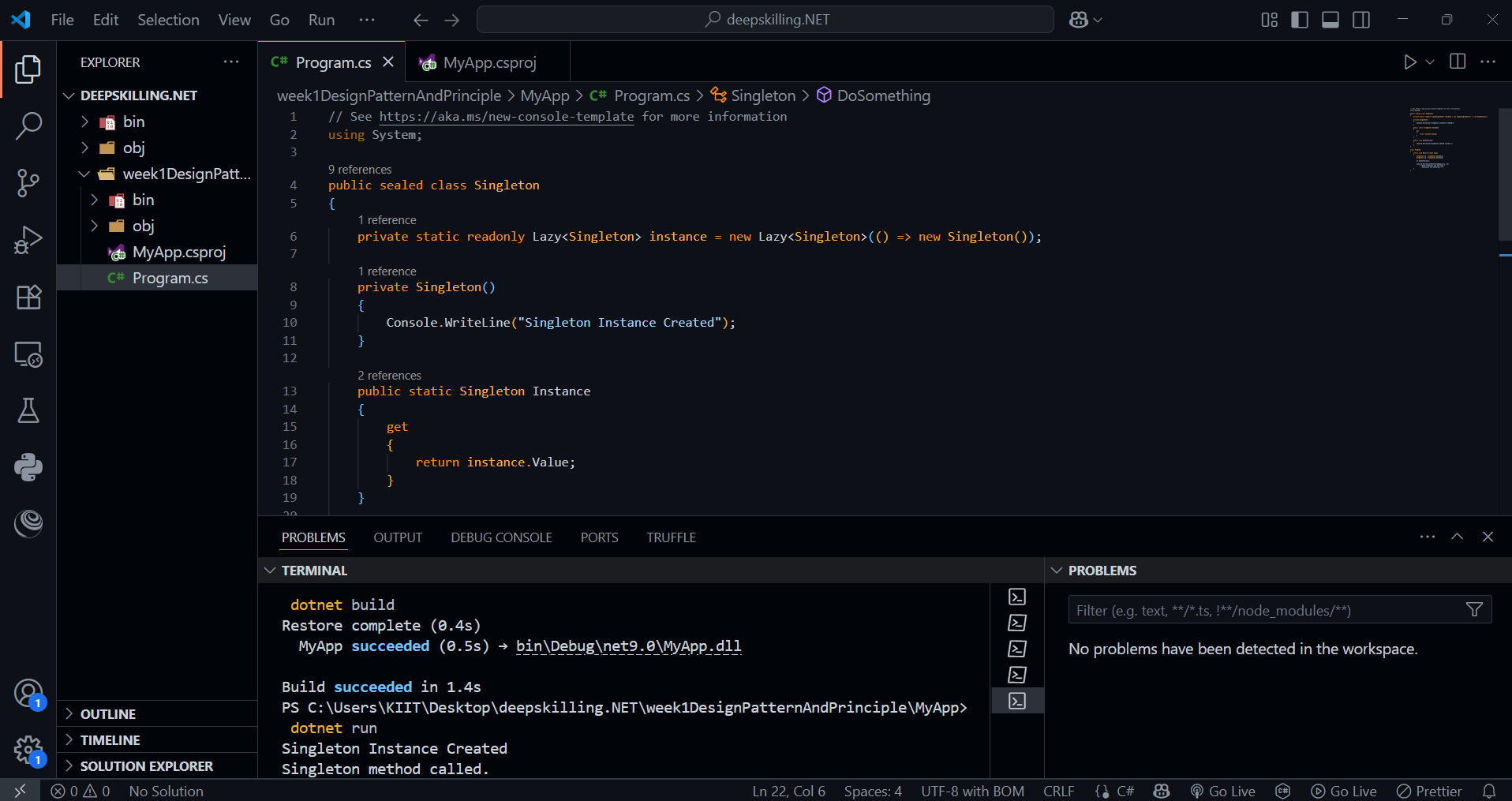
            ? "Both instances are the same."

            : "Instances are different.");

    }

}

Output



Exercise 2: Implementing the Factory Method Pattern

using System;

namespace FactoryMethodPattern

{

    // Product Interface

    public interface ITransport

    {

        void Deliver();

    }

    // Concrete Product 1

    public class Truck : ITransport

    {

        public void Deliver()

        {

            Console.WriteLine("Delivery by land in a truck.");

        }

    }

    // Concrete Product 2

    public class Ship : ITransport

    {

        public void Deliver()

        {

            Console.WriteLine("Delivery by sea in a ship.");

        }

    }

    // Creator (Factory)

    public abstract class Logistics

    {

        // Factory Method

        public abstract ITransport CreateTransport();

        public void PlanDelivery()

        {

            ITransport transport = CreateTransport();

            transport.Deliver();

        }

    }

    // Concrete Creator 1

    public class RoadLogistics : Logistics

    {

        public override ITransport CreateTransport()

        {

            return new Truck();

        }

    }

    // Concrete Creator 2

    public class SeaLogistics : Logistics

    {

        public override ITransport CreateTransport()

        {

            return new Ship();

        }

    }

    // Client

    class Program

    {

        static void Main(string[] args)

        {

            Logistics roadLogistics = new RoadLogistics();

            roadLogistics.PlanDelivery();

            Logistics seaLogistics = new SeaLogistics();

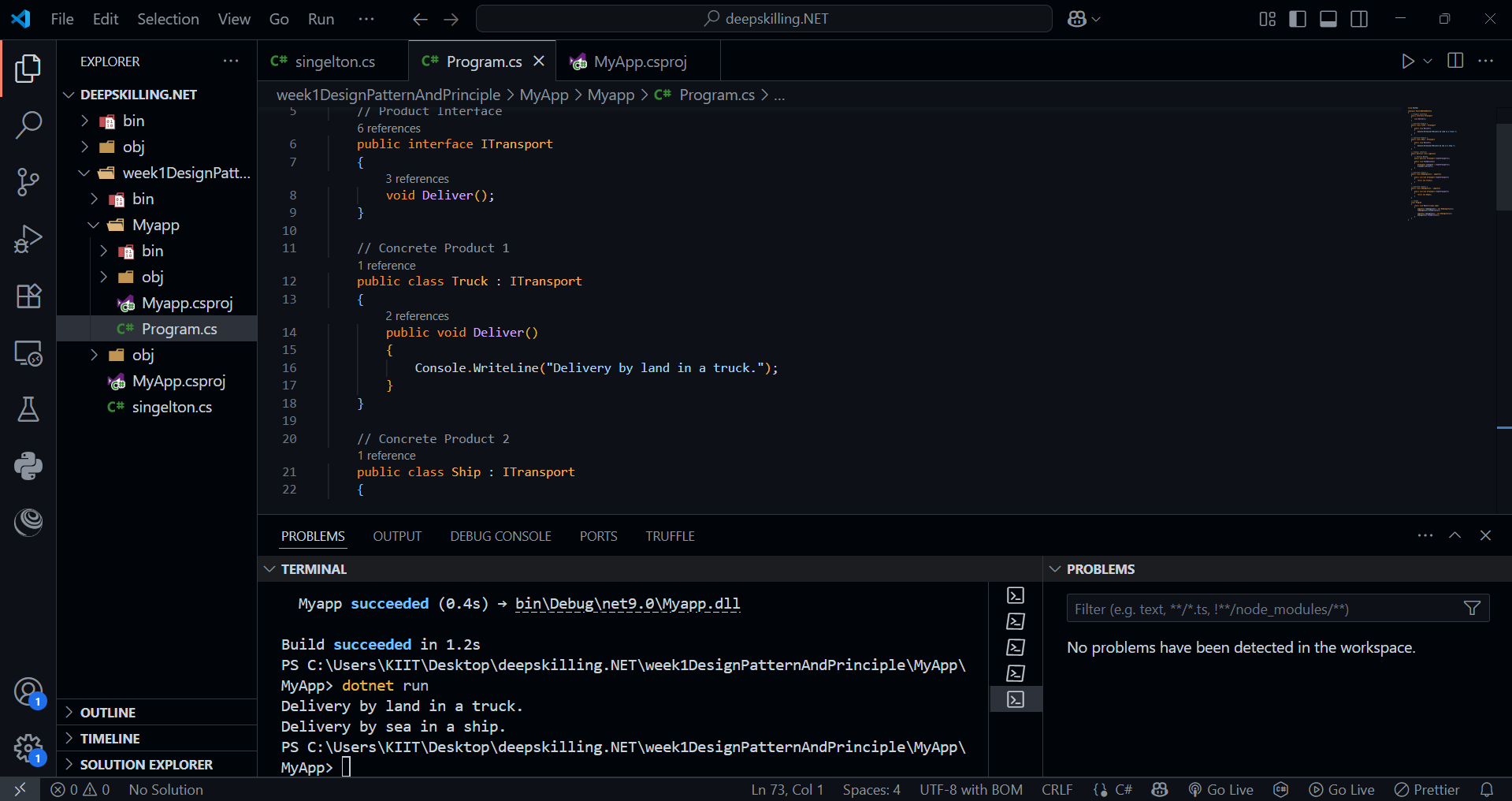
            seaLogistics.PlanDelivery();

        }

    }

}

Output



Module 2 - Data Structures and Algorithms

Exercise 2: E-commerce Platform Search Function

using System;

using System.Collections.Generic;

using System.Linq;

namespace ECommerceSearch

{

    // Product class

    public class Product

    {

        public int Id { get; set; }

        public string Name { get; set; }

        public string Category { get; set; }

        public double Price { get; set; }

        public override string ToString()

        {

            return $"[{Id}] {Name} - {Category} - ₹{Price}";

        }

    }

    // Product catalog with search functionality

    public class ProductCatalog

    {

        private List<Product> products = new List<Product>();

        public ProductCatalog()

        {

            // Sample data

            products.Add(new Product { Id = 1, Name = "Wireless Mouse", Category = "Electronics", Price = 799.00 });

            products.Add(new Product { Id = 2, Name = "Bluetooth Speaker", Category = "Electronics", Price = 1299.00 });

            products.Add(new Product { Id = 3, Name = "Yoga Mat", Category = "Fitness", Price = 499.00 });

            products.Add(new Product { Id = 4, Name = "Running Shoes", Category = "Footwear", Price = 1999.00 });

            products.Add(new Product { Id = 5, Name = "Water Bottle", Category = "Fitness", Price = 299.00 });

        }

        // Search function

        public List<Product> Search(string keyword)

        {

            return products

                .Where(p => p.Name.Contains(keyword, StringComparison.OrdinalIgnoreCase) ||

                            p.Category.Contains(keyword, StringComparison.OrdinalIgnoreCase))

                .ToList();

        }

    }

    class Program

    {

        static void Main(string[] args)

        {

            var catalog = new ProductCatalog();

            Console.WriteLine("🔍 Welcome to E-Shop! Search for a product:");

            string keyword = Console.ReadLine();

            var results = catalog.Search(keyword);

            Console.WriteLine("\n📦 Search Results:");

            if (results.Count == 0)

            {

                Console.WriteLine("No matching products found.");

            }

            else

            {

                foreach (var product in results)

                {

                    Console.WriteLine(product);

                }

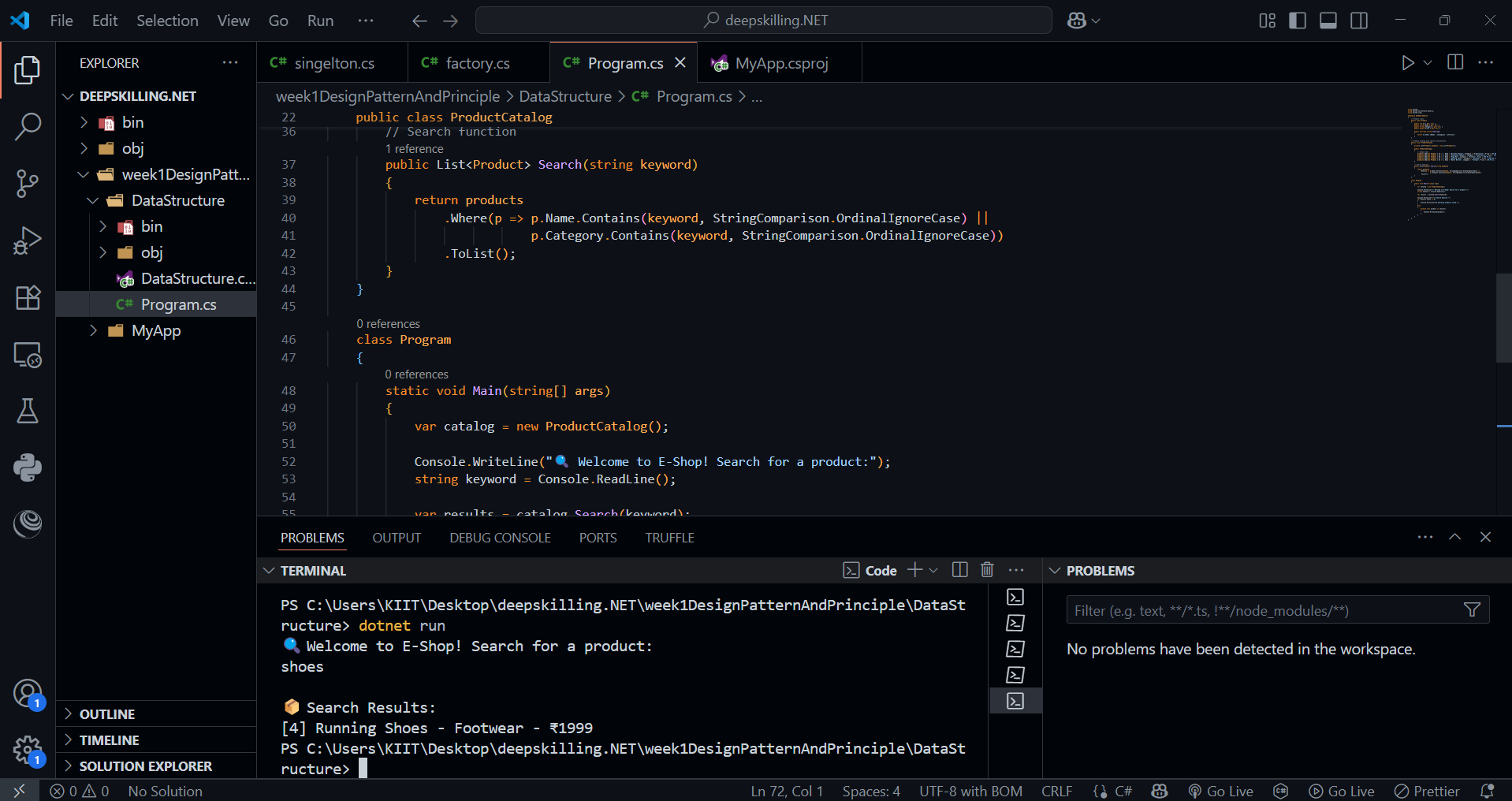
            }

        }

    }

}

Output



Exercise 7: Financial Forecasting

using System;

class Program

{

    static void Main(string[] args)

    {

        Console.WriteLine("📊 Financial Forecasting Tool");

        Console.Write("Enter Initial Investment (₹): ");

        double principal = Convert.ToDouble(Console.ReadLine());

        Console.Write("Enter Monthly Contribution (₹): ");

        double monthlyContribution = Convert.ToDouble(Console.ReadLine());

        Console.Write("Enter Annual Interest Rate (%): ");

        double annualRate = Convert.ToDouble(Console.ReadLine()) / 100;

        Console.Write("Enter Forecast Period (years): ");

        int years = Convert.ToInt32(Console.ReadLine());

        int n = 12; // Compounded monthly

        double totalMonths = years \* 12;

        double amount = principal;

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

        {

            // Apply monthly compound interest

            amount = amount \* (1 + annualRate / n) + monthlyContribution;

        }

        Console.WriteLine($"\n📈 Forecast after {years} years:");

        Console.WriteLine($"Total Value: ₹{Math.Round(amount, 2)}");

        Console.WriteLine($"Total Invested: ₹{Math.Round(principal + monthlyContribution \* totalMonths, 2)}");

        Console.WriteLine($"Total Gain: ₹{Math.Round(amount - (principal + monthlyContribution \* totalMonths), 2)}");

    }

}

Output

