1) **SingletonPatternExample**

Logger.cs

using System;

public sealed class Logger

{

private static Logger? \_instance = null;

private static readonly object \_lock = new object();

private Logger()

{

Console.WriteLine("Logger instance created.");

}

public static Logger Instance

{

get

{

if (\_instance == null)

{

lock (\_lock)

{

if (\_instance == null)

{

\_instance = new Logger();

}

}

}

return \_instance;

}

}

public void Log(string message)

{

Console.WriteLine($"[LOG] {message}");

}

}

Program.cs

using System;

class Program

{

static void Main()

{

Console.WriteLine("Testing Singleton Logger:\n");

Logger logger1 = Logger.Instance;

logger1.Log("This is the first log message.");

Logger logger2 = Logger.Instance;

logger2.Log("This is the second log message.");

Console.WriteLine($"\nlogger1 and logger2 refer to the same instance: {ReferenceEquals(logger1, logger2)}");

}

}

2) **FactoryMethodPatternExample**

IDocument.cs

public interface IDocument

{

void Open();

}

WordDocument.cs

using System;

public class WordDocument : IDocument

{

public void Open()

{

Console.WriteLine("Opening a Word document...");

}

}

PdfDocument.cs

using System;

public class PdfDocument : IDocument

{

public void Open()

{

Console.WriteLine("Opening a PDF document...");

}

}

ExcelDocument.cs

using System;

public class ExcelDocument : IDocument

{

public void Open()

{

Console.WriteLine("Opening an Excel document...");

}

}

DocumentFactory.cs

public abstract class DocumentFactory

{

public abstract IDocument CreateDocument();

}

WordDocumentFactory.cs

public class WordDocumentFactory : DocumentFactory

{

public override IDocument CreateDocument()

{

return new WordDocument();

}

}

PdfDocumentFactory.cs

public class PdfDocumentFactory : DocumentFactory

{

public override IDocument CreateDocument()

{

return new PdfDocument();

}

}

ExcelDocumentFactory.cs

public class ExcelDocumentFactory : DocumentFactory

{

public override IDocument CreateDocument()

{

return new ExcelDocument();

}

}

Program.cs

using System;

class Program

{

static void Main()

{

DocumentFactory wordFactory = new WordDocumentFactory();

IDocument wordDoc = wordFactory.CreateDocument();

wordDoc.Open();

DocumentFactory pdfFactory = new PdfDocumentFactory();

IDocument pdfDoc = pdfFactory.CreateDocument();

pdfDoc.Open();

DocumentFactory excelFactory = new ExcelDocumentFactory();

IDocument excelDoc = excelFactory.CreateDocument();

excelDoc.Open();

}

}

3) **E-commerce Platform Search Function**

Product.cs

﻿public class ProductAdd commentMore actions

{

public int ProductId { get; set; }

public string ProductName { get; set; }

public string Category { get; set; }

public Product(int id, string name, string category)

{

ProductId = id;

ProductName = name;

Category = category;

}

public override string ToString()

{

return $"{ProductId}: {ProductName} ({Category})";

}Add commentMore actions

}

Program.cs  
using System;Add commentMore actions

class Program

{

static Product LinearSearch(Product[] products, string name)

{

foreach (var product in products)

{

if (product.ProductName.Equals(name, StringComparison.OrdinalIgnoreCase))

return product;

}

return null;

}

static Product BinarySearch(Product[] products, string name)

{

int left = 0, right = products.Length - 1;

while (left <= right)

{

int mid = (left + right) / 2;

int comparison = string.Compare(products[mid].ProductName, name, true);

if (comparison == 0)

return products[mid];

else if (comparison < 0)

left = mid + 1;

else

right = mid - 1;

}

return null;

}

static void Main()

{

var products = new Product[]

{

new Product(1, "Laptop", "Electronics"),

new Product(2, "Shoes", "Fashion"),

new Product(3, "Phone", "Electronics"),

new Product(4, "Watch", "Accessories"),

new Product(5, "Bag", "Fashion")

};

Console.WriteLine("Enter product name to search:");

string searchName = Console.ReadLine();

var result1 = LinearSearch(products, searchName);

Console.WriteLine("\n[Linear Search]");

Console.WriteLine(result1 != null ? result1.ToString() : "Product not found.");

Array.Sort(products, (a, b) => string.Compare(a.ProductName, b.ProductName, true));

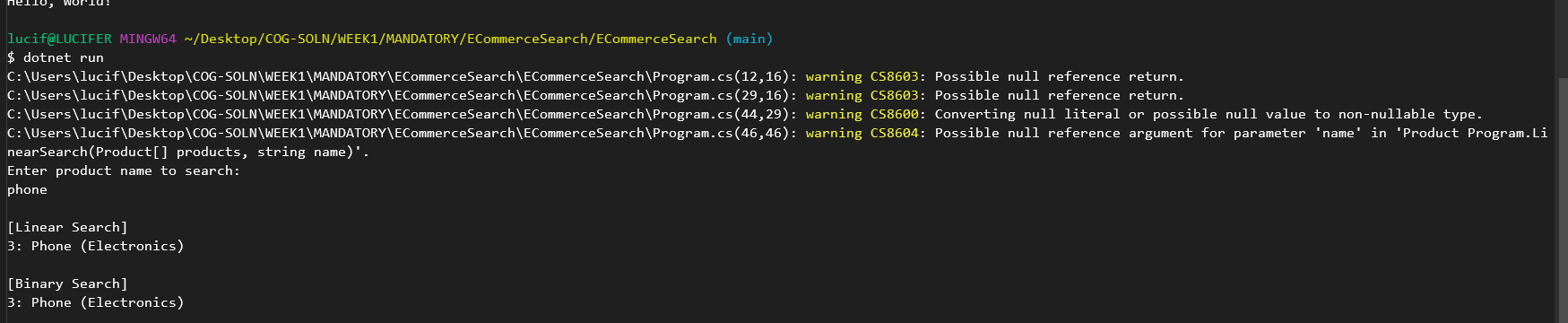
var result2 = BinarySearch(products, searchName);

Console.WriteLine("\n[Binary Search]");

Console.WriteLine(result2 != null ? result2.ToString() : "Product not found.");

}

}



4) **Financial Forecasting**

Program.cs

using System;Add commentMore actions

class Program

{

static void Main()

{

double currentValue = 1000;

double growthRate = 0.10;

int years = 5;

Console.WriteLine("=== Financial Forecasting Tool ===\n");

Console.WriteLine("Input Summary:");

Console.WriteLine($"- Starting Value : {currentValue:C}");

Console.WriteLine($"- Annual Growth Rate : {growthRate \* 100}%");

Console.WriteLine($"- Forecast Period : {years} year(s)\n");

double forecast = ForecastValueRecursive(currentValue, growthRate, years);

Console.WriteLine("Forecast Result:");

Console.WriteLine($"Predicted value after {years} years: {forecast:C}");

}

static double ForecastValueRecursive(double value, double rate, int years)

{

if (years == 0)

return value;

return ForecastValueRecursive(value \* (1 + rate), rate, years - 1);

}

}

