**Exercise 1: Implementing the Singleton Pattern**

using System;

public class Logger

{

    private static Logger instance;

    private static readonly object lockObj = new object();

    private Logger() {}

    public static Logger GetInstance()

    {

        if (instance == null)

        {

            lock (lockObj)

            {

                if (instance == null)

                    instance = new Logger();

            }

        }

        return instance;

    }

    public void Log(string message) => Console.WriteLine("Log: " + message);

}

public class Program

{

    public static void Main()

    {

        var logger1 = Logger.GetInstance();

        var logger2 = Logger.GetInstance();

        logger1.Log("Message 1");

        logger2.Log("Message 2");

        Console.WriteLine("Same instance: " + (logger1 == logger2));

    }

}

**Output:-**

**Screenshot 2025-06-19 225912**

**Exercise 2: Implementing the Factory Method Pattern**

using System;

public interface IDocument { void Open(); }

public class WordDocument : IDocument { public void Open() => Console.WriteLine("Opening Word Document"); }

public class PdfDocument : IDocument { public void Open() => Console.WriteLine("Opening PDF Document"); }

public class ExcelDocument : IDocument { public void Open() => Console.WriteLine("Opening Excel Document"); }

public abstract class DocumentFactory { public abstract IDocument CreateDocument(); }

public class WordFactory : DocumentFactory { public override IDocument CreateDocument() => new WordDocument(); }

public class PdfFactory : DocumentFactory { public override IDocument CreateDocument() => new PdfDocument(); }

public class ExcelFactory : DocumentFactory { public override IDocument CreateDocument() => new ExcelDocument(); }

public class Program

{

    public static void Main()

    {

        DocumentFactory factory = new WordFactory();

        factory.CreateDocument().Open();

        factory = new PdfFactory();

        factory.CreateDocument().Open();

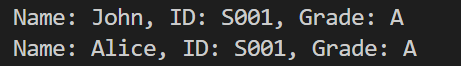
        factory = new ExcelFactory();

        factory.CreateDocument().Open();

    }

}

**Output:-**

****

**Exercise 3: Implementing the Builder Pattern**

using System;

public class Computer

{

    public string CPU { get; private set; }

    public string RAM { get; private set; }

    public string Storage { get; private set; }

    private Computer() { }

    public class Builder

    {

        private readonly Computer computer = new Computer();

        public Builder SetCPU(string cpu) { computer.CPU = cpu; return this; }

        public Builder SetRAM(string ram) { computer.RAM = ram; return this; }

        public Builder SetStorage(string storage) { computer.Storage = storage; return this; }

        public Computer Build() => computer;

    }

}

public class Program

{

    public static void Main()

    {

        var pc = new Computer.Builder().SetCPU("Intel i9").SetRAM("32GB").SetStorage("1TB SSD").Build();

        Console.WriteLine($"CPU={pc.CPU}, RAM={pc.RAM}, Storage={pc.Storage}");

    }

}

**Output:-**

**Screenshot 2025-06-19 225752**

**Exercise 4: Implementing the Adapter Pattern**

using System;

public interface IPaymentProcessor { void ProcessPayment(); }

public class OldGateway { public void MakePayment() => Console.WriteLine("OldGateway payment processed"); }

public class NewGateway { public void CompleteTransaction() => Console.WriteLine("NewGateway payment processed"); }

public class OldGatewayAdapter : IPaymentProcessor

{

    private readonly OldGateway old = new OldGateway();

    public void ProcessPayment() => old.MakePayment();

}

public class NewGatewayAdapter : IPaymentProcessor

{

    private readonly NewGateway nw = new NewGateway();

    public void ProcessPayment() => nw.CompleteTransaction();

}

public class Program

{

    public static void Main()

    {

        IPaymentProcessor oldAdapter = new OldGatewayAdapter();

        oldAdapter.ProcessPayment();

        IPaymentProcessor newAdapter = new NewGatewayAdapter();

        newAdapter.ProcessPayment();

    }

}

**Output:-**

**Screenshot 2025-06-19 225712**