**Exercise 1: Implementing the Singleton Pattern**

**Code**

ANS) using System;

namespace SingletonExample

{

public sealed class Singleton

{

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

// Private constructor to prevent instantiation

private Singleton()

{

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

}

// Public accessor

public static Singleton Instance => \_instance.Value;

// Sample method

public void DoSomething()

{

Console.WriteLine("Singleton is working.");

}

}

class Program

{

static void Main(string[] args)

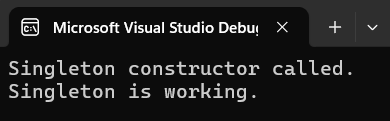
{

Singleton.Instance.DoSomething();

}

}

}



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

**Code**

ANS) using System;

// Product Interface

public interface IDocument

{

void Print();

}

// Concrete Product 1

public class Resume : IDocument

{

public void Print()

{

Console.WriteLine("Printing Resume Document.");

}

}

// Concrete Product 2

public class Report : IDocument

{

public void Print()

{

Console.WriteLine("Printing Report Document.");

}

}

// Creator (Factory)

public abstract class DocumentCreator

{

public abstract IDocument CreateDocument();

}

// Concrete Creator 1

public class ResumeCreator : DocumentCreator

{

public override IDocument CreateDocument()

{

return new Resume();

}

}

// Concrete Creator 2

public class ReportCreator : DocumentCreator

{

public override IDocument CreateDocument()

{

return new Report();

}

}

// Client Code

class Program

{

static void Main(string[] args)

{

DocumentCreator creator;

// Create Resume

creator = new ResumeCreator();

IDocument doc1 = creator.CreateDocument();

doc1.Print();

// Create Report

creator = new ReportCreator();

IDocument doc2 = creator.CreateDocument();

doc2.Print();

}

}

