**DoFactory Proxy Design Pattern**

**Notes: -**

**1-proxy design pattern: provide a surrogate or placeholder for another object to control access to it.**

**Example: -**

**namespace DoFactoryProxyPatternPro.Subjects{**

**abstract class Subject{public abstract void Request();}}**

**using System;**

**namespace DoFactoryProxyPatternPro.Subjects{**

**class RealSubject : Subject{**

**public override void Request(){**

**Console.WriteLine("Called RealSubject.Request()");}}}**

**namespace DoFactoryProxyPatternPro.Subjects{**

**//this class contain field that assign to another child class that inherit from Subject abstract class**

**class Proxy : Subject{**

**private RealSubject \_realSubject;**

**public override void Request(){**

**// Use 'lazy initialization'**

**if (\_realSubject == null){**

**\_realSubject = new RealSubject();}**

**\_realSubject.Request();}}}**

**using DoFactoryProxyPatternPro.Subjects;**

**using static System.Console;**

**namespace DoFactoryProxyPatternPro{**

**//Provide a surrogate or placeholder for another object to control access to it.**

**class Program{**

**static void Main(string[] args){**

**// Create proxy and request a service**

**Proxy proxy = new Proxy();**

**proxy.Request();**

**ReadKey();}}}**

**Example: -**

**namespace DoFactoryProxy2Pro.Models.Interfaces{**

**public interface IMath{**

**double Add(double x, double y);**

**double Sub(double x, double y);**

**double Mul(double x, double y);**

**double Div(double x, double y);}}**

**using DoFactoryProxy2Pro.Models.Interfaces;**

**namespace DoFactoryProxy2Pro.Models{**

**class Math : IMath{**

**public double Add(double x, double y) { return x + y; }**

**public double Sub(double x, double y) { return x - y; }**

**public double Mul(double x, double y) { return x \* y; }**

**public double Div(double x, double y) { return x / y; }}}**

**using DoFactoryProxy2Pro.Models.Interfaces;**

**namespace DoFactoryProxy2Pro.Models{**

**class MathProxy : IMath{**

**private Math \_math = new Math();**

**public double Add(double x, double y){return \_math.Add(x, y);}**

**public double Sub(double x, double y){return \_math.Sub(x, y);}**

**public double Mul(double x, double y){return \_math.Mul(x, y);}**

**public double Div(double x, double y){return \_math.Div(x, y);}}}**

**using DoFactoryProxy2Pro.Models;**

**using static System.Console;**

**namespace DoFactoryProxy2Pro{**

**class Program{**

**static void Main(string[] args){**

**// Create math proxy**

**MathProxy proxy = new MathProxy();**

**// Do the math**

**WriteLine("4 + 2 = " + proxy.Add(4, 2));**

**WriteLine("4 - 2 = " + proxy.Sub(4, 2));**

**WriteLine("4 \* 2 = " + proxy.Mul(4, 2));**

**WriteLine("4 / 2 = " + proxy.Div(4, 2));**

**ReadKey();}}}**