**DoFactory Composite Design Pattern**

**Notes:-**

**1- Compose objects into tree structures to represent part-whole hierarchies. Composite lets clients treat individual objects and compositions of objects uniformly.**

**(By making base class and make sub class derived from the base class and contains property of list of the base class, so its allow us to create single or composite object to insert it)**

**namespace DoFactorySecondPro.Models{**

**abstract class DrawingElement{**

**protected string \_name;**

**public DrawingElement(string name){this.\_name = name;}**

**public abstract void Add(DrawingElement d);**

**public abstract void Remove(DrawingElement d);**

**public abstract void Display(int indent);}}**

**using System;**

**using System.Collections.Generic;**

**namespace DoFactorySecondPro.Models{**

**//this class contains List<DrawingElement> which allow us to add single or group of base class**

**class CompositeElement : DrawingElement{**

**private List<DrawingElement> elements = new List<DrawingElement>();**

**public CompositeElement(string name): base(name){}**

**public override void Add(DrawingElement d){elements.Add(d);}**

**public override void Remove(DrawingElement d){elements.Remove(d);}**

**public override void Display(int indent){**

**Console.WriteLine(new String('-', indent) + "+ " + \_name);**

**foreach (DrawingElement d in elements){d.Display(indent + 2);}}}}**

**using System;**

**namespace DoFactorySecondPro.Models{**

**class PrimitiveElement : DrawingElement{**

**public PrimitiveElement(string name): base(name){}**

**public override void Add(DrawingElement c){**

**Console.WriteLine("Cannot add to a PrimitiveElement");}**

**public override void Remove(DrawingElement c){**

**Console.WriteLine("Cannot remove from a PrimitiveElement");}**

**public override void Display(int indent){**

**Console.WriteLine(**

**new String('-', indent) + " " + \_name);}}}**

**using DoFactorySecondPro.Models;**

**using System;**

**namespace DoFactorySecondPro{**

**class Program{**

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

**// Create a tree structure**

**CompositeElement root =**

**new CompositeElement("Picture");**

**root.Add(new PrimitiveElement("Red Line"));**

**root.Add(new PrimitiveElement("Blue Circle"));**

**root.Add(new PrimitiveElement("Green Box"));**

**// Create a branch**

**CompositeElement comp =**

**new CompositeElement("Two Circles");**

**comp.Add(new PrimitiveElement("Black Circle"));**

**comp.Add(new PrimitiveElement("White Circle"));**

**root.Add(comp);**

**// Add and remove a PrimitiveElement**

**PrimitiveElement pe =**

**new PrimitiveElement("Yellow Line");**

**root.Add(pe);**

**root.Remove(pe);**

**root.Display(1);**

**Console.ReadKey();}}}**

**Example:-**

**We make abstract class and derived two class one primitive and other is composite that contains list of the abstract class to allow store single or group of the same class type**

**namespace DoFactoryFirstPro.Models{**

**abstract class Component{**

**protected string name;**

**public Component(string name){this.name = name;}**

**public abstract void Add(Component c);**

**public abstract void Remove(Component c);**

**public abstract void Display(int depth);}}**

**using System;**

**using System.Collections.Generic;**

**namespace DoFactoryFirstPro.Models{**

**class Composite : Component{**

**private List<Component> \_children = new List<Component>();**

**public Composite(string name): base(name){ }**

**public override void Add(Component component){\_children.Add(component);}**

**public override void Remove(Component component){\_children.Remove(component);}**

**public override void Display(int depth){**

**Console.WriteLine(new String('-', depth) + name);**

**foreach (Component component in \_children){**

**component.Display(depth + 2);}}}}**

**using System;**

**namespace DoFactoryFirstPro.Models{**

**class Leaf : Component{**

**public Leaf(string name): base(name){}**

**public override void Add(Component c){Console.WriteLine("Cannot add to a leaf");}**

**public override void Remove(Component c){Console.WriteLine("Cannot remove from a leaf");}**

**public override void Display(int depth){Console.WriteLine(new String('-', depth) + name);}}}**

**using DoFactoryFirstPro.Models;**

**using static System.Console;**

**namespace DoFactoryFirstPro{**

**class Program{**

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

**// Create a tree structure**

**Composite root = new Composite("root");**

**root.Add(new Leaf("Leaf A"));**

**root.Add(new Leaf("Leaf B"));**

**Composite comp = new Composite("Composite X");**

**comp.Add(new Leaf("Leaf XA"));**

**comp.Add(new Leaf("Leaf XB"));**

**//we store group of Composite instead of one item**

**root.Add(comp);**

**root.Add(new Leaf("Leaf C"));**

**// Add and remove a leaf**

**Leaf leaf = new Leaf("Leaf D");**

**root.Add(leaf);**

**root.Remove(leaf);**

**// Recursively display tree**

**root.Display(1);**

**ReadKey();}}}**