**Section19 Strategy Design Pattern**

**Notes: -**

**1-Strategy pattern: Define a family of algorithms, encapsulate each one, and make them interchangeable. Strategy lets the algorithm vary independently from clients that use it.**



**(by making instance of the Concrete Strategy as parameter on the Context class we can access to the function we want to the target Concrete Strategy as below)**

**Example: -**

**namespace DoFactoryStrategySol.Strategies{**

**abstract class Strategy{**

**public abstract void AlgorithmInterface();}}**

**using System;**

**namespace DoFactoryStrategySol.Strategies{**

**class ConcreteStrategyA : Strategy{**

**public override void AlgorithmInterface(){**

**Console.WriteLine("Called ConcreteStrategyA.AlgorithmInterface()");}}}**

**using System;**

**namespace DoFactoryStrategySol.Strategies{**

**class ConcreteStrategyB : Strategy{**

**public override void AlgorithmInterface(){**

**Console.WriteLine("Called ConcreteStrategyB.AlgorithmInterface()");}}}**

**using System;**

**namespace DoFactoryStrategySol.Strategies{**

**class ConcreteStrategyC : Strategy{**

**public override void AlgorithmInterface(){**

**Console.WriteLine("Called ConcreteStrategyC.AlgorithmInterface()");}}}**

**using DoFactoryStrategySol.Strategies;**

**namespace DoFactoryStrategySol{**

**class Context{**

**private Strategy \_strategy;**

**public Context(Strategy strategy){this.\_strategy = strategy;}**

**public void ContextInterface(){\_strategy.AlgorithmInterface();}}}**

**using DoFactoryStrategySol.Strategies;**

**using System;**

**namespace DoFactoryStrategySol{**

**class Program{**

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

**Context context;**

**// Three contexts following different strategies**

**context = new Context(new ConcreteStrategyA());**

**context.ContextInterface();**

**context = new Context(new ConcreteStrategyB());**

**context.ContextInterface();**

**context = new Context(new ConcreteStrategyC());**

**context.ContextInterface();**

**// Wait for user**

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

**Example: -**

**//this example provide register multiple values and set different strategies as below**

**using System.Collections.Generic;**

**namespace DoFactoryStrategy2Pro.Strategies{**

**abstract class SortStrategy{public abstract void Sort(List<string> list);}}**

**using System;**

**using System.Collections.Generic;**

**namespace DoFactoryStrategy2Pro.Strategies{**

**class QuickSort : SortStrategy{**

**public override void Sort(List<string> list){**

**list.Sort();**

**Console.WriteLine("QuickSorted list ");}}}**

**using System;**

**using System.Collections.Generic;**

**namespace DoFactoryStrategy2Pro.Strategies{**

**class ShellSort : SortStrategy{**

**public override void Sort(List<string> list){Console.WriteLine("ShellSorted list ");}}}**

**using System;**

**using System.Collections.Generic;**

**namespace DoFactoryStrategy2Pro.Strategies{**

**class MergeSort : SortStrategy{**

**public override void Sort(List<string> list){Console.WriteLine("MergeSorted list ");}}}**

**using DoFactoryStrategy2Pro.Strategies;**

**using System;**

**using System.Collections.Generic;**

**namespace DoFactoryStrategy2Pro{**

**//this class we apply SortedList which provide the functionality of attach strategy class and call it**

**class SortedList{**

**private List<string> \_list = new List<string>();**

**private SortStrategy \_sortstrategy;**

**public void SetSortStrategy(SortStrategy sortstrategy){this.\_sortstrategy = sortstrategy;}**

**public void Add(string name){\_list.Add(name);}**

**public void Sort(){\_sortstrategy.Sort(\_list);**

**foreach (string name in \_list){Console.WriteLine(" " + name);}**

**Console.WriteLine();}}}**

**using DoFactoryStrategy2Pro.Strategies;**

**using System;**

**namespace DoFactoryStrategy2Pro{**

**class Program{**

**//we make register multiple list and apply multiple strategies as below**

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

**SortedList studentRecords = new SortedList();**

**studentRecords.Add("Samual");**

**studentRecords.Add("Jimmy");**

**studentRecords.Add("Sandra");**

**studentRecords.Add("Vivek");**

**studentRecords.Add("Anna");**

**studentRecords.SetSortStrategy(new QuickSort());**

**studentRecords.Sort();**

**studentRecords.SetSortStrategy(new ShellSort());**

**studentRecords.Sort();**

**studentRecords.SetSortStrategy(new MergeSort());**

**studentRecords.Sort();**

**// Wait for user**

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