**Lesson03 what is Functional Builder Design Pattern**

**Notes:-**

**1-the idea of the functional builder pattern is to provide build multiple actions through the action / delegates to build the instance of class properties**

**(We see that in the functional builder you register all the functions and then in the Builder method all the actions are executed once you called it, unlike the fluent builder that execute each method separated not in specific method call like build)**

**Steps:-**

**// create the model class called Person**

**public class Person{**

**public string Name { get; set; }**

**public string Position { get; set; }}**

**//this class is Builder that we use Action as delegate to refer to the input Person and output**

**public sealed class PersonBuilder{**

**public readonly List<Action<Person>> Actions = new List<Action<Person>>();**

**//actions here is delegate that have input of type Person and it will set property Name value**

**public PersonBuilder Called(string name){**

**Actions.Add(p => { p.Name = name; });**

**return this;}**

**//actions here is delegate that will set the property Area to the Person instance**

**public PersonBuilder SetArea(string area){**

**Actions.Add(p => { p.Area = area; });**

**return this;}**

**//this method call actions that will apply the anonymous method and it will set**

**//all the properties for the Person class**

**public Person Build(){**

**var p = new Person();**

**Actions.ForEach(a => a(p));**

**return p;}}**

**//this extension method it will apply Person Builder with pass the Actions to set property //to the Person instance**

**public static class PersonBuilderExtensions{**

**//this extension method contain add the actions which refer to the person instance**

**public static PersonBuilder WorksAsA(this PersonBuilder builder, string position){**

**builder.Actions.Add(p =>{p.Position = position;});**

**return builder;}}**

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

**//we see that we call chain method of functional builder to set multiple properities**

**var pb = new PersonBuilder();**

**var person = pb.Called("Dmitri").SetArea("Area 1").WorksAsA("Programmer").Build();**

**Console.WriteLine($"{person.Name} - {person.Area} - {person.Position}");**

**Console.ReadLine();}**