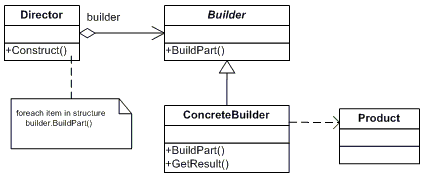
**Lesson05 dofactory Builder Design Pattern**

**Notes:-**

**1-Builder Design Pattern: its Separate the construction of a complex object from its representation so that the same construction process can create different representations.**



**In the below example we see that the director contains the call functions we want to implement the builder not contains chain call of methods like previous lessons**

**public class Product{**

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

**public void Add(string part){\_parts.Add(part);}**

**public void Show(){**

**Console.WriteLine("\nProduct Parts -------");**

**foreach (string part in \_parts)**

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

**public abstract class Builder{**

**public abstract void BuildPartA();**

**public abstract void BuildPartB();**

**public abstract Product GetResult();}**

**//each builder inherit from the Builder abstract class and implement its way**

**public class ConcreteBuilder1 : Builder{**

**private Product \_product = new Product();**

**public override void BuildPartA(){\_product.Add("PartA");}**

**public override void BuildPartB(){\_product.Add("PartB");}**

**public override Product GetResult(){return \_product;}}**

**//each builder inherit from the Builder abstract class and implement its way**

**public class ConcreteBuilder2 : Builder{**

**private Product \_product = new Product();**

**public override void BuildPartA(){\_product.Add("PartX");}**

**public override void BuildPartB(){\_product.Add("PartY");}**

**public override Product GetResult(){return \_product;}}**

**//we pass the abstract class as parameter on the Director that used on the main method , we see //that we don’t have chain method call as below**

**public class Director{**

**// Builder uses a complex series of steps**

**public void Construct(Builder builder){**

**builder.BuildPartA();**

**builder.BuildPartB();}}**

**public static void Main(){**

**// Create director and builders**

**Director director = new Director();**

**var b1 = new ConcreteBuilder1();**

**var b2 = new ConcreteBuilder2();**

**// Construct two products**

**director.Construct(b1);**

**Product p1 = b1.GetResult();**

**p1.Show();**

**director.Construct(b2);**

**Product p2 = b2.GetResult();**

**p2.Show();**

**Console.ReadKey();}**