**Lesson04 what is Facade Builder Design Pattern**

**Notes:-**

**1-this is the most famous way of the other three way standard and fluent and functional builder , this way provide façade builder that provide access to the sub builder by provide instance of each builder and on each sub builder class method it will return the class itself to provide chain of calling functions as below**

**Steps:-**

**1-on the Person.cs we set the following code as below**

**public class Person{**

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

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

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

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

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

**public int AnnualIncome { get; set; }**

**public override string ToString(){**

**return $"{nameof(StreetAddress)}: {StreetAddress},{nameof(PostCode)}: {PostCode},{nameof(City)}: {City},{nameof(CompanyName)}: {CompanyName},{nameof(Position)}: {Position},{nameof(AnnualIncome)}: {AnnualIncome}";}}**

**2-on the PersonBuilder.cs we define the façade builder**

**//is facade for other builder , it allow access to sub builder**

**//in order to provide access to child builder we initialize instance to each builder**

**public class PersonBuilder{**

**//reference**

**protected Person person = new Person();**

**public PersonJobBuilder works => new PersonJobBuilder(person);**

**public PersonAddressBuilder Address => new PersonAddressBuilder(person);**

**//provide static method that provide impolicit conversion from PersonBuilder to Person**

**public static implicit operator Person(PersonBuilder pb){return pb.person;}}**

**3-on the PersonJobBuilder class it inherit from the façade builder**

**//on the child class it inherit from the base class and person**

**public class PersonJobBuilder : PersonBuilder{**

**public PersonJobBuilder(Person person){this.person = person;}**

**//it will return the class instance to apply Builder pattern**

**public PersonJobBuilder At(string companyName){**

**person.CompanyName = companyName;**

**return this;}**

**//it will return the class instance to apply Builder pattern**

**public PersonJobBuilder AsA(string position){**

**person.Position = position;**

**return this;}**

**//it will return the class instance to apply Builder pattern**

**public PersonJobBuilder Earning(int amount){**

**person.AnnualIncome = amount;**

**return this;}}**

**4-on the PersonAddressBuilder.cs it inherit from the façade builder**

**//on the child class it inherit from the base class and person**

**public class PersonAddressBuilder : PersonBuilder{**

**public PersonAddressBuilder(Person person){this.person = person;}**

**//it will return the class instance to apply Builder pattern**

**public PersonAddressBuilder At(string streetAddress){**

**person.StreetAddress = streetAddress;**

**return this;}**

**//it will return the class instance to apply Builder pattern**

**public PersonAddressBuilder WithPostCode(string postCode){**

**person.PostCode = postCode;**

**return this;}**

**//it will return the class instance to apply Builder pattern**

**public PersonAddressBuilder In(string city){**

**person.City = city;**

**return this;}}**

**5-on the program.cs we set the following code**

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

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

**//it will internally call implicit conversion PersonBuilder to Person**

**Person personBuilder = pb.works.At("Fabrikana").AsA("Developer").Earning(2000)**

**.Address.At("AliBaba").WithPostCode("POB").In("Jordan");**

**WriteLine(personBuilder.ToString());**

**ReadLine();}**