Abstract Factory Pattern Implementation Lab Task 1

Introduction

In this lab task we will learn how to implement the abstract factory pattern using C#.Net

- Abstract Factory is a creational design pattern that lets you produce families of related objects without specifying their concrete classes.
- In this pattern, you provide a way to encapsulate a group of individual factories that have a common theme.
- This pattern helps you to interchange specific implementations without changing the code that uses them, even at runtime. However, it may result in unnecessary complexity and extra work.
- An abstract factory is called a factory of factories.

Implementation

1. Create a console application in visual studio.
2. Create a folder and name it AbstractFactoryImplementation
3. Create following classes files in this folder:

   a. AbstractProductA.cs

   b. ConcreteProductA1.cs

   c. ConcreteProductB1.cs

   d. AbstractProductB.cs

   e. ConcreteProductA2.cs

   f. ConcreteProductB2.cs

In each file above, visual studio automatically will create classes with the name same as the file name.

Up to now, in this demo we have accomplished a task of creating families of related product classes that is the fundamental requirement for abstract factory design pattern problem scenarios.

Aamir Shabbir Pare

In the next step we are going to create abstract factory and concrete factories in which the related products will be created as prescribed by the pattern.

4. Create following files in the same folder:

   a. AbstractFactory.cs

   b. ConcreteFactoryA.cs

   b. ConcreteFactoryB.cs

5. Write the following code in each of the above files:

AbstractFactory.cs

```
public interface AbstractFactory
{
    AbstractProductA CreateProductA();
    AbstractProductB CreateProductB();
}
```

ConcreteFactoryA.cs

```
public class ConcreteFactoryA : AbstractFactory
{
    public AbstractProductA CreateProductA()
    {
        Console.WriteLine("ProductA1 is created in ConcreteFactoyA");
        return new ProductA1();
    }
```

```
    public AbstractProductB CreateProductB()
    {
        Console.WriteLine("ProductB1 is created in ConcreteFactoyA");

        return new ProductB1();
    }
}
```

ConcreteFactoryB.cs

```
public class ConcreteFactoryB : AbstractFactory
{
    public AbstractProductA CreateProductA()
    {
        Console.WriteLine("ProductA2 is created in ConcreteFactoyB");
        return new ProductA2();
    }

    public AbstractProductB CreateProductB()
    {
        Console.WriteLine("ProductB2 is created in ConcreteFactoyB");

        return new ProductB2();
    }
}
```

Up to now we have finished the task of creating abstract factory and related concrete factories used to create the related products.

Next, we are going to create a client that uses the abstract factory to create products without knowing the details how and where the products are going to be created.

6. Create a Client.cs file in the same folder and write the below code in it.

```
public class Client
{
    public AbstractFactory Factory { set; get; }
    public Client(AbstractFactory fac)
    {
        Factory = fac;
    }
}
```

7. Create DemoAbstractFactory.cs file and write the write the following code in it. Here we are using the above client in Main method to test the above implemented abstract factory pattern.

```
public class DemoAbstractFactory
{
    public static void Main(string[] args)
    {
        var clientA = new Client(new ConcreteFactoryA());
        var productA1 = clientA.Factory.CreateProductA();
        var prodcutB1 = clientA.Factory.CreateProductB();

        var clientB = new Client(new ConcreteFactoryB());
        var productA2 = clientB.Factory.CreateProductA();
        var prodcutB2 = clientB.Factory.CreateProductB();
```


```
        Console.ReadKey();
    }
}
```

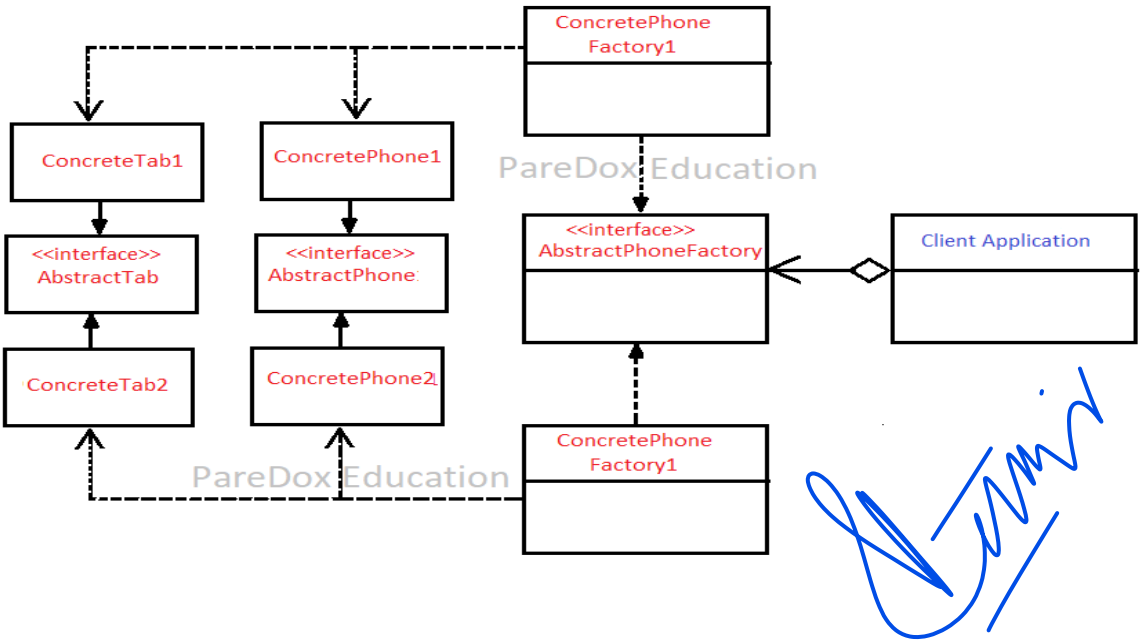
8 In visual studio press f5 to execute the demo.

==========END TASK==========

Abstract Factory Pattern Implementation Lab Task 2

Now you have grabbed the basic understanding of the abstract factory design pattern and how it solves the problems that occur in software design. You have also learned the implantation of this patterns.

It's time to bother your brain to look at UML diagram given below and provide its implantation at you own.



Goodbye, wish you all the best and see you in next lab task!

Aamir Shabbir Pare