Yazılım Tasarımı ve Mimarisi Dersi

Factory (Fabrika) Deseni Deney Föyü

Hazırlık Soruları:

1. Factory deseni nedir, ne amaçla kullanılır, nasıl kodlanır?
2. Abstract factory deseni hangi probleme çözüm olarak tasarlanmıştır.

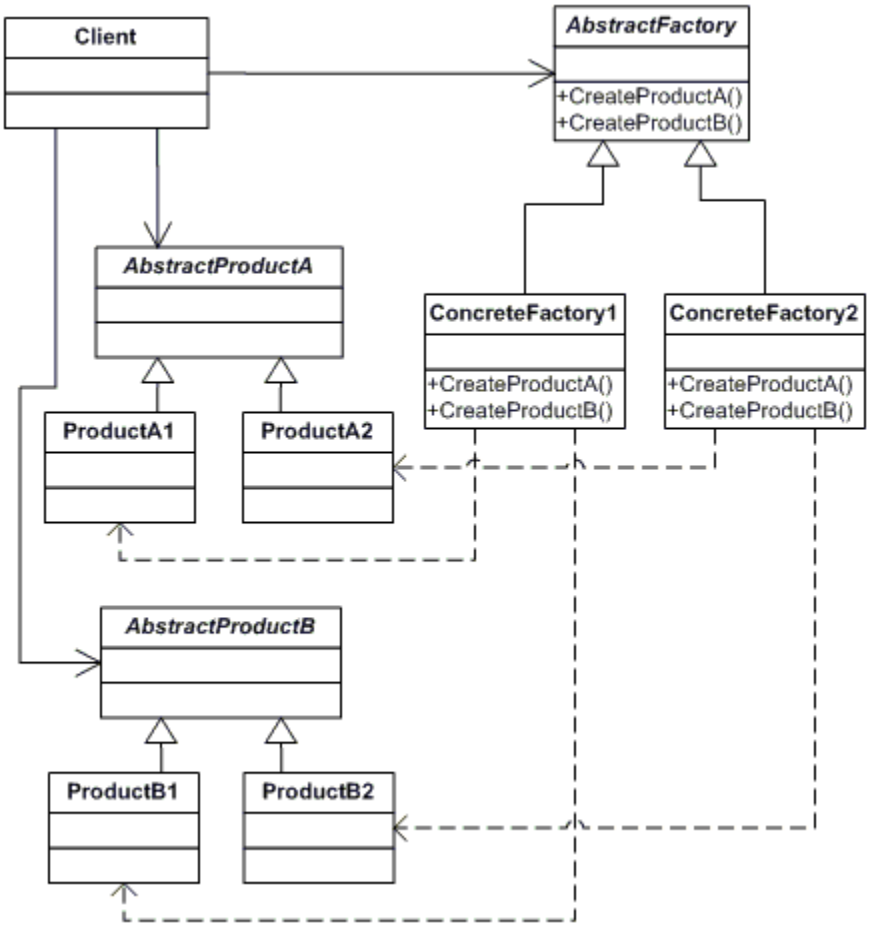
Giriş

- Amacı bağımlı ya da ilgili ürün aileleri için ortak bir arayüz sağlamaktır.

Bu desen sayesinde istemci fabrikalardan da soyutlanır. Yani istemcinin, hangi fabrikada ürünün üretileceğini bilmesine gerek olmaz.

Doğrudan ürün ile iletişime geçmez. Bu sayede daha sonra istemcide değişiklik yapmak zorunda kalınmaz.

UML



SoyutFabrika (AbstractFactory)

Ürün aileleri için ortak arayüz. Somut fabrikalar bu arayüzü implement ederler.

SomutFabrika (ConcreteFactory)

Somut ürün nesnelerinin üretildiği fabrika nesneleridir.

SoyutÜrün (AbstractProduct)

Ürün tipleri için ortak arayüz

Ürün (Product)

Ürün nesnesini ifade eder.

SoyutÜrün arayüzünü implement eden somut fabrika ile oluşturulur.

İstemci (Client)

SoyutFabrika ve SoyutÜrün tarafından bildirilen arayüzleri kullanan sınıf.

Yapısal Kod:

```
using System;


namespace ConsoleApplication7
{
    class Program
    {
        static void Main(string[] args)
        {
            // Abstract factory #1
            SoyutFabrika factory1 = new SomutFabrika_1();
            Istemci client1 = new Istemci(factory1);
            client1.Run();

            // Abstract factory #2
            SoyutFabrika factory2 = new SomutFabrika_2();
            Istemci client2 = new Istemci(factory2);
            client2.Run();


            Console.ReadKey();
        }
    }
}

    abstract class SoyutFabrika
    {
        public abstract SoyutUrun_A UrunOlustur_A();
        public abstract SoyutUrun_B UrunOlustur_B();
    }


    class SomutFabrika_1 : SoyutFabrika
    {
        public override SoyutUrun_A UrunOlustur_A()
        {
```

```
            return new Urun_A1();
        }
        public override SoyutUrun_B UrunOlustur_B()
        {
            return new Urun_B1();
        }
    }


    class SomutFabrika_2 : SoyutFabrika
    {
        public override SoyutUrun_A UrunOlustur_A()
        {
            return new Urun_A2();
        }
        public override SoyutUrun_B UrunOlustur_B()
        {
            return new Urun_B2();
        }
    }


    abstract class SoyutUrun_A
    {
    }


    abstract class SoyutUrun_B
    {
    }

    class Urun_A1 : SoyutUrun_A
    {
    }

    class Urun_B1 : SoyutUrun_B
    {

    }

    class Urun_A2 : SoyutUrun_A
    {
    }

    class Urun_B2 : SoyutUrun_B
    {

    }

    class Istemci
    {
        private SoyutUrun_A abstractProductA;
        private SoyutUrun_B abstractProductB;

        // Constructor
        public Istemci(SoyutFabrika factory)
        {
            abstractProductB = factory.UrunOlustur_B();
            abstractProductA = factory.UrunOlustur_A();
        }
    }
```

Soru 1:

Yukarıdaki yapısal kodu abstract sınıf yerine interface kullanarak yeniden düzenleyiniz.

Yukarıdaki yapısal koda 3. bir ürün ailesi eklenmek istenirse oluşacak durumu kodlayınız.

Örnek Uygulama:

```
//Sekiller için bir arayuz
public interface Sekiller
{
    void ciz();
}

//Sekiller arayuzunu implemen eden urunler icin ola
public class Dikdortgen : Sekiller
{
    public void ciz()
    {
        Console.WriteLine("Inside Dikdortgen::draw() method.");
    }
}

public class Kare : Sekiller
{
    public void ciz()
    {
        Console.WriteLine("Inside Kare::draw() method.");
    }
}

public class Cember : Sekiller
{

    public void ciz()
    {
        Console.WriteLine("Inside Cember::draw() method.");
    }
}


public interface Renkler
{
    void boya();
}


public class Kirmizi : Renkler
{

    public void boya()
    {
        Console.WriteLine("Inside Kirmizi::boya() method.");
    }
}


public class Yesil : Renkler
```

```
{

    public void boya()
    {
        Console.WriteLine("Inside Yesil::boya() method.");
    }
}


public class Mavi : Renkler
{

    public void boya()
    {
        Console.WriteLine("Inside Mavi::boya() method.");
    }
}

public abstract class SoyutFabrikam
{
    public abstract Renkler rengiGetir(String renk);
    public abstract Sekiller sekliGetir(String sekil);
}


public class SekilFabrikasi : SoyutFabrikam
{
    public override Renkler rengiGetir(string renk)
    {
        throw new NotImplementedException();
    }

    public override Sekiller sekliGetir(String sekilTipi)
    {

        if (sekilTipi == null)
        {
            return null;
        }

        if (sekilTipi == "CEMBER")
        {
            return new Cember();

        }
        else if (sekilTipi == "DIKDORTGEN")
        {
            return new Dikdortgen();

        }
        else if (sekilTipi == "KARE")
        {
            return new Kare();
        }

        return null;
    }

}

public class ColorFactory : SoyutFabrikam
{
```

```
    public override Sekiller sekliGetir(String sekilTipi)
    {
        return null;
    }


    public override Renkler rengiGetir(String renk)
    {

        if (renk == null)
        {
            return null;
        }

        if (renk == "KIRMIZI")
        {
            return new Kirmizi();

        }
        else if (renk == "GREEN")
        {
            return new Yesil();

        }
        else if (renk == "MAVI")
        {
            return new Mavi();
        }

        return null;
    }
}


public class FactoryProducer
{
    public static SoyutFabrikam getFactory(String secim)
    {

        if (secim == "SEKIL")
        {
            return new SekilFabrikasi();

        }
        else if (secim == "RENK")
        {
            return new ColorFactory();
        }

        return null;
    }
}

public class AbstractFactoryPatternDemo
{
    public static void main(String[] args)
    {

        //Sekil fabrikasini uret
        SoyutFabrikam shapeFactory = FactoryProducer.getFactory("SEKIL");
```

```
        //get an object of Shape Cember
        Sekiller shape1 = shapeFactory.sekliGetir("CEMBER");

        //call draw method of Shape Cember
        shape1.ciz();

        //get an object of Shape Dikdortgen
        Sekiller shape2 = shapeFactory.sekliGetir("DIKDORTGEN");

        //call draw method of Shape Dikdortgen
        shape2.ciz();

        //get an object of Shape Kare
        Sekiller shape3 = shapeFactory.sekliGetir("KARE");

        //call draw method of Shape Kare
        shape3.ciz();

        //get color factory
        SoyutFabrikam colorFactory = FactoryProducer.getFactory("RENK");

        //get an object of Color Kirmizi
        Renkler color1 = colorFactory.rengiGetir("KIRMIZI");

        //call boya method of Kirmizi
        color1.boya();

        //get an object of Color Yesil
        Renkler color2 = colorFactory.rengiGetir("YESIL");

        //call boya method of Yesil
        color2.boya();

        //get an object of Color Mavi
        Renkler color3 = colorFactory.rengiGetir("MAVI");

        //call boya method of Color Mavi
        color3.boya();
    }
}
```