Yazılım Tasarımı ve Mimarisi Dersi

Factory (Fabrika) Deseni Deney Föyü

Hazırlık Soruları:

1. Interface, abstract class nedir, ne amaçla kullanılır, farkları ve ortak yanları nelerdir?
2. Coupling ve Cohesion kavramlarını açıklayınız.
3. Low coupling principle nedir, açıklayınız?
4. Factory deseni nedir, hangi probleme çözüm olarak kullanılmaktadır.

Giriş

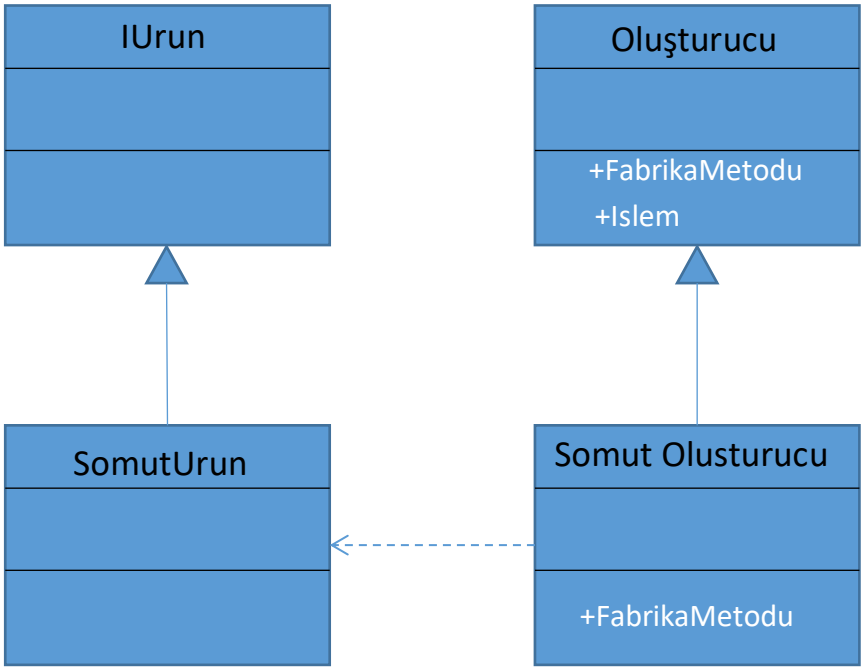
- İstemcinin her nesne hakkında detaylı bilgiye sahip olması ve bu bilgi ile tüm nesnelerin oluşum sürecine dahil olması gerekmez ve istenmez.
- Bu amaçla oluşturulacak benzer tipte nesneler ortak bir arayüze bağlanır.
- İhtiyaca yönelik nesneler bir oluşturucu sınıf aracılığı ile oluşturulur.
- Bu sayede istemci, oluşturulmasını istediği ürünü(nesneyi), nasıl üretildiğini bilmeksizin; sadece istediği ürünü parametre olarak belirterek üretilmesini sağlar.

İstemci ürünün üretilmesinden soyutlanmış olur.

Bağımlılık azalır.

Tekrar kullanılabilirlik artar.

UML



Sınıflar Hakkında Bilgi:

IUrun (Product):

Fabrika metodunun ürettiği nesneler için ortak arayüz tanımlar.

SomutUrun (ConcreteProduct) :

Urun arayüzünü implement eden sınıf. Urunler bu sınıf ile tanımlanır.

Olusturucu (Creator):

Urun tipinde nesne döndüren fabrika metodunun bildirildiği yerdir. Varsayılan bir durum implement edilebilir burada.

SomutOlusturucu (ConcreteCreator):

Fabrika metodunun hazırlandığı ya da override edilerek SomutUrun nesnesinin döndürüldüğü kısımdır.

Kalıp:

```
interface Urun
    {

    }

    class Urun_A : Urun
    {
    }

    class Urun_B : Urun
    {
    }

    abstract class Olusturucu
    {
        public abstract Urun FabrikaMetodu(string type);
    }

    class SomutOlusturucu : Olusturucu
    {
        public override Urun FabrikaMetodu(string type)
        {
            switch (type)
            {
                case "A": return new Urun_A();
                case "B": return new Urun_B();
                default: throw new ArgumentException("Invalid type", "type");
            }
        }
    }
```

Örnek:

```
public interface Compressing
{
    void CompressThis();
}

public class Zip : Compressing
{
    public void CompressThis()
    {
        Console.WriteLine("Olusturulan dosya tipi \".zip\"");
    }
}

public class Rar : Compressing
{

    public void CompressThis()
    {
        Console.WriteLine("Olusturulan dosya tipi \".rar\"");
    }
}


public class Tar : Compressing
{

    public void CompressThis()
    {
        Console.WriteLine("Olusturulan dosya tipi \".tar\"");
    }
}

public abstract class CompressFactory
{
    public abstract Compressing getType(string ct);
}

public class SomutCompressFactory
{

    public Compressing getType(String compressType)
    {
        if (compressType == null)
        {
            return null;
        }
        if (compressType.Equals("ZIP"))
        {
            return new Zip();

        }
        else if (compressType.Equals("RAR"))
        {
            return new Rar();

        }
        else if (compressType.Equals("TAR"))
        {
```

```
            return new Tar();
        }

        return null;
    }
}


public class FactoryPatternDemo
{

    public static void main(String[] args)
    {
        SomutCompressFactory cFactory = new SomutCompressFactory();

        Compressing comp1 = cFactory.getType("RAR");
        comp1.CompressThis();


        Compressing comp2 = cFactory.getType("ZIP");
        comp2.CompressThis();


        Compressing comp3 = cFactory.getType("TAR");
        comp3.CompressThis();
    }
}
```