**EBMYL512**

**Nesneye Yönelik Yazılım Mühendisliği Dersi Ödevi**

**2017-2018 Bahar Dönemi**

1. **Template pattern** tasarım şablonuyla ilgili soruları cevaplayınız.
2. Bu tasarım şablonunun kullanılmasındaki motivasyon nedir?

Template pattern tasarım kalıbı daha çok sıralı operasyonları içeren fonksiyonellikleri ilgilendirmektedir. Öyle ki bu fonksiyonellikler içerisine dahil olan operasyonların bazıları, duruma göre farklı şekillerde uygulanmak istenebilir. Dolayısıyla fonksiyonelliğin sahiplendiği ve çalışma biçimleri değişkenlik gösterebilecek olan operasyonların, kolayca genişletilebilmesi ve sahip oldukları kod parçalarının yeniden kullanılabilirliğinin arttırılması noktasında bir çözüm gerekmektedir. Template pattern şablonu burada çözüm olarak kullanılabilir.

1. Bu tasarım şablonu hakkında detaylı bilgi veriniz.

Template pattern kalıbı davranışsal tasarım kalıpları grubunda yer alır. Bu tasarım şablonu bir işlem için gerekli adımları soyut olarak tanımlar ve bir şablon metot ile algoritmanın nasıl çalışacağını belirler. Alt sınıflar algoritma için gerekli bir ya da birden fazla metodu kendi bünyelerinde gerçekleyerek, kullanılan algoritmanın kendi istekleri doğrultusunda çalışmasını sağlar. Böylece hem kod tekrarının önüne geçilerek kodun tekrar kullanılabilirliği hem de algoritma iskeletinde yapılacak bir düzenlemenin tek bir yerden yapılması sağlanır.

Template pattern şablonunda bir operasyonu gerçekleştirmek için gerekli olan adımlar ve bu adımları çalıştıracak olan metotlar abstract sınıfta tanımlanır. Bu abstract sınıfı uygulayan gerçek kümeler gerekli adımları overwrite eder ve bu adımları kullanan metotlar çağırılarak işlem gerçekleştirilir.

1. Bir örnekle kullanımını açıklayınız.

Örnekte "SicakIcecek" parent class içerisinde bir "Hazirla" şablon metodu hazırlanmış, çay ve nescafe içecekleri için gerekli yerlerde değişiklik yaparak kullanılmıştır. Bu şablon içinde tanımlanmış "SuKaynat" ve "BardagaDoldur" fonksiyonlarının tüm içecekler için aynı olması nedeniyle bu fonksiyonlar geçersiz kılınması isteğe bağlı olacak şekilde şekilde yani **virtual** olarak tanımlanmıştır. "Icerik" fonksiyonu ise çay ve nescafe için farklılık göstereceğinden bu fonksiyon tanımlanması zorunlu olacak şekilde **abstract**olarak tanımlanmıştır. Çay ve Nescafe alt sınıflarında "Icerik" farklı olarak tanımlanmış ve daha sonra "Hazirla" metodu çağırılarak program çıktıları elde edilmiştir. Bu durumda programa yeni bir içecek eklenmesi gerektiğinde bütün kodları kopyalamak yerine yeni içecek sınıfını "SicakIcecek" parent class türetip "Icerik" metodunu tanımlamak yeterli olur. Böylece hem kod tekrarının önüne geçilmiş hem de sade ve anlaşılır bir kod yazılmış olur.

using System;  
  
namespace SablonMetotKalibi  
{  
    public abstract class SicakIcecek  
    {  
        public virtual void SuKaynat()  
        {  
            Console.WriteLine("Su kaynatıldı.");  
        }  
  
        // PrimitiveOperation  
        public abstract void Icerik();  
  
        public virtual void BardagaDoldur()  
        {  
            Console.WriteLine("İçeçek bardağa dolduruldu.");  
        }  
  
        // TemplateMethod  
        public void Hazirla()  
        {  
            SuKaynat();  
            Icerik();  
            BardagaDoldur();  
            Console.WriteLine("İçeçek hazırlandı.");  
        }  
    }  
      
    public class Cay : SicakIcecek  
    {      
        public override void Icerik()  
        {  
            Console.WriteLine("Çay demlendi.");  
        }  
    }  
      
    public class Nescafe : SicakIcecek  
    {  
        public override void Icerik()  
        {  
            Console.WriteLine("Nescafe ve süt tozu konuldu.");  
        }  
    }  
  
    class Program  
    {  
        static void Main(string[] args)  
        {  
            SicakIcecek cay = new Cay();  
            cay.Hazirla();  
  
            Console.WriteLine();  
  
            SicakIcecek nescafe = new Nescafe();  
            nescafe.Hazirla();  
  
            Console.ReadLine();  
        }  
    }  
}

1. Bu tasarım şablonunu kullanan örnek bir program veriniz.
2. Sorumlu olduğunuz AntiPattern (**Dead End**, **Architecture by Implication**, **Irrational Management**) ile ilgili aşağıdaki soruları cevaplayınız
3. Bu AntiPattern’i ortaya çıkaran root cause nedir?

* **Dead End**: Bu AntiPattern’i ortaya çıkaran kök sebep olarak bileşendeki iyileştirmeler yardımıyla ürünlerin veya sonraki sürümlerinin yeniden kullanıma sunulması gösterebilir.
* **Architecture by Implication**: Bu AntiPattern’i ortaya çıkaran tipik nedenleri risk yönetiminin olmayışı, yöneticilerin, mimarların veya geliştiricilerin aşırı ilgisi, kritik bölgelerde farklılık gösterebilen önceki deneyimlere güvenme, sistem mühendisliğindeki boşlukların neden olduğu örtük ve çözümlenmemiş mimarlık sorunları olarak sıralayabiliriz. Buradan kök neden olarak gurur ve tembellik çıkarımını yapabiliriz.
* **Irrational Management**: Bu AntiPattern’i ortaya çıkaran tipik nedenleri yöneticinin ehil olmayışı, bu nedenle vizyonu ve stratejisinin olmaması, karar verme ve uygulama kabiliyetinin eksikliği, başarısızlık korkusu olarak sıralanabilir. Buradan kök neden olarak sorumluluk çıkarılabilir.

1. Bu AntiPattern’e verilen başka isimler varsa nelerdir?

* **Dead End** a.k.a: Kevorkian Component
* **Architecture by Implication** a.k.a: Wherefore art thou architecture?
* **Irrational Management** a.k.a: Pathological Supervisor, Short-Term Thinking, Managing by Reaction, Decision Phobia, Managers Playing with Technical Toys

1. Bu AntiPattern hangi problem sınıfına bir çözüm önermektedir?

* **Dead End**: Bu AntiPattern, tahliye kodu gibi temel araştırmaları destekleyen test yataklarında kabul edilebilir bir çözüm olabilir ve özelleştirme ile önemli faydalar sağlanır.
* **Architecture by Implication**: Bu AntiPatten, geliştirilmekte olan bir sistem için mimari özelliklerin eksikliğini karakterize eder.
* **Irrational Management**: Bu AntiPattern, yöneticilerin personel yeteneklerini, hem güçlü hem de zayıf yanlarını tanıma, personel becerilerine uygun hedefler sağlama, çalışanlarla etkili iletişim kurma gibi sorunlarını esas alır

1. Bu AntiPattern hakkında detaylı bilgi veriniz.

* **Dead End**: Değiştirilen bileşen artık tedarikçi tarafından desteklenmediği takdirde, yeniden kullanılabilir bir bileşenin değiştirilmesine Dead End. Bu değişiklikler yapıldığında , destek yükü uygulama sistemi geliştiricilerine ve bakım verenlere aktarılır. Yeniden kullanılabilir bileşendeki iyileştirmeler kolaylıkla entegre edilemez ve değişiklik konusunda destek sorunları suçlanabilir.
* **Architecture by Implication**: Implication AntiPattern tarafından mimariye odaklı yeniden yapılandırılmış çözüm, sistem mimarisi tanımına düzenli bir yaklaşım getirmekte ve sistemin çoklu görünümlerine dayanmaktadır. Her bir görüş sistemi, sistem paydaşının perspektifinden modeller. Her paydaş, yüksek öncelikli soru ve sorunlardan sorumludur ve her bir görüş, tüm bilgi sistemini temsil eder. Bu durum önemli soruları ve sorunları yanıtlar.
* **Irrational Management**: Bu AntiPattern, projeyi yürüten insanların kişiliklerine kadar izlenebilecek bir dizi yaygın yazılım projesi sorununu kapsar. Irrational Management, bir yöneticinin geliştirme personelini doğrudan yönetememesiyle önemli ölçüde birleşmiştir. Bu aynı zamanda, yetersizlik tarafından karakterize edilen iyi bir insan-yönetim becerisi eksikliği olarak adlandırılır.