**T.C.**

**MANİSA CELÂL BAYAR ÜNİVERSİTESİ**

**TEKNOLOJİ FAKÜLTESİ**

**YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**

**YZM 2118 YAZILIM MİMARİSİ VE TASARIMI  
Ödev Dokümantasyonu**

**Fatih ÖZKIR**

**Mayıs – 2017**

|  |
| --- |
| 1. **Soru 1** |
| * 1. Verilen UML Diyagramının Coupling, Cohesion, SRP Ve OCP Tasarım Prensipleri Açısından Yorumlanması   2. Tasarımsal Açıdan İyileştirilmiş UML Diyagramı   3. Tasarım Class ve Client Kodları |

1. **Soru 1**

# **1.1 Verilen UML Diyagramının Coupling, Cohesion, SRP Ve OCP Tasarım Prensipleri Açısından Yorumlanması**

# 

### **- UML Diyagramı –**

Verilen UML diyagramını **Coupling (Bağlılık)** ve **Cohesion (Yapışıklık)** açısından yorumlayacak olursak;

Player sınıfının **Resim** ve **Video** classlarına **yüksek coupling ve düşük cohesion** özelliği söz konusudur ki **Resim** ve **Video** sınıflarında yapılan bir değişiklik doğrudan olarak **Player** sınıfını da etkileyecektir ve bundan dolayı **Player** sınıfı içinde de güncelleme, düzenleme gibi işlemler yapmak zorunda kalınacaktır. Bunlar haricinde başka oynatılabilir bir nesne geldiğinde de **Player** class’ı içerisine yeni gelen nesnenin **property(özellik)** olarak eklenmesi gerekecektir.

Bu tip yakından ilişkili class tasarımları yapılırken mümkün mertebede classlar arasında bir izolasyon olması gerekmektedir ve bu izolasyon sayesinde **düşük coupling** **yüksek cohesion** özelliği sağlanarak daha **esnek** bir tasarıma sahip olunmuş olunur.

Şimdi de verilen UML diyagramını **SRP (Single Responsibility Principle (Tek Sorumluk Tasarım Prensibi))** ile yorumlayalım;

Bu prensibe göre yorum yapmak için Player class’ı üzerinden ilerleyelim. Player class’ı üzerinde sonradan yapılacak bir değişiklik durumunda class üzerinde yapısal bir değişiklik söz konusu olabilmektedir. Player sınıfındaki kodlar tekrar kullanılmak istendiğinde, Player sınıfının bağımlı olduğu sınıflarda bu sınıfla beraber tekrardan kullanılmak zorunda kalınacaktır ve bu duruma göre çıkarım gerçekleştirecek olursak Player sınıfı kullanılmak istediği yere beraberinde kullandığı diğer sınıfları da götürecektir ki bu da kodun tekrar kullanımını zorlaştıracaktır.

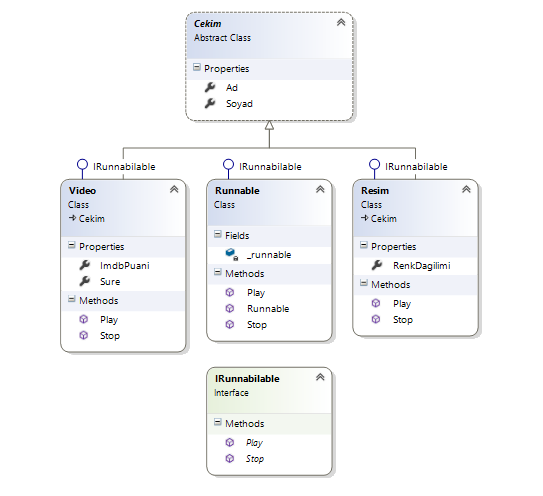
Son olarak verilen UML diyagramını **OCP (Open Closed Principle (Açık Kapalı Tasarım Prensibi))** ile yorumlayalım;

Yine Player class’ını kullanacak olursak, Player class’ı Resim ve Video classlarını kullanarak **Play** ve **Stop** operasyonlarını gerçekleştirmektedir. Eğer bu operasyonlara sahip yeni bir class ekleneceğini düşünürsek Player class’ı içerisine yeni gelen class’ı da eklemek zorunda kalacağız ki bu durum sürekli Player class’ı içerinde yapısal bir değişikliğe sebep olacaktır.

# **1.2 Tasarımsal Açıdan İyileştirilmiş UML Diyagramı**



**- Tasarımsal Açıdan İyileştirilmiş UML Class Diyagramı –**

****

**- Tasarımsal Açıdan İyileştirilmiş Visual Studio Class Diyagram –**

# **1.2 Tasarım Class ve Client Kodları**

public abstract class Cekim public interface IRunnabilable

{ {

public string Ad { get; set; } void Play();

public string CekimYapan { get; set; } void Stop();

} }

public class Video : Cekim, IRunnabilable public class Resim : Cekim, IRunnabilable

{ {

public uint Sure { get; set; } public string RenkDagilimi { get; set; }

public float ImdbPuani { get; set; }

public void Play() public void Play()

{ {

Console.Write("Video Oynatılıyor.\n"); Console.Write("Resimler Oynatılıyor.\n");

} }

public void Stop() public void Stop()

{ {

Console.Write("Video Durduruldu.\n"); Console.Write("Resimler Durduruldu.\n");

} }

} }

public class Runnable:IRunnabilable class Program

{ {

private IRunnabilable \_runnable; static void Main(string[] args)

public Runnable(IRunnabilable runnable) {

{ Runnable resimRunnable=new Runnable(new Resim());

\_runnable = runnable; //IRunnabilable resimRunnable=new Resim(); resimRunnable.Play();

} resimRunnable.Stop();

public void Play() Runnable videoRunnable=new Runnable(new Video());

{ //IRunnabilable videoRunnable=new Video();

\_runnable.Play(); videoRunnable.Play();

} videoRunnable.Stop();

public void Stop() Console.Read();

{ }

\_runnable.Stop(); }

}

}