Adapter Pattern

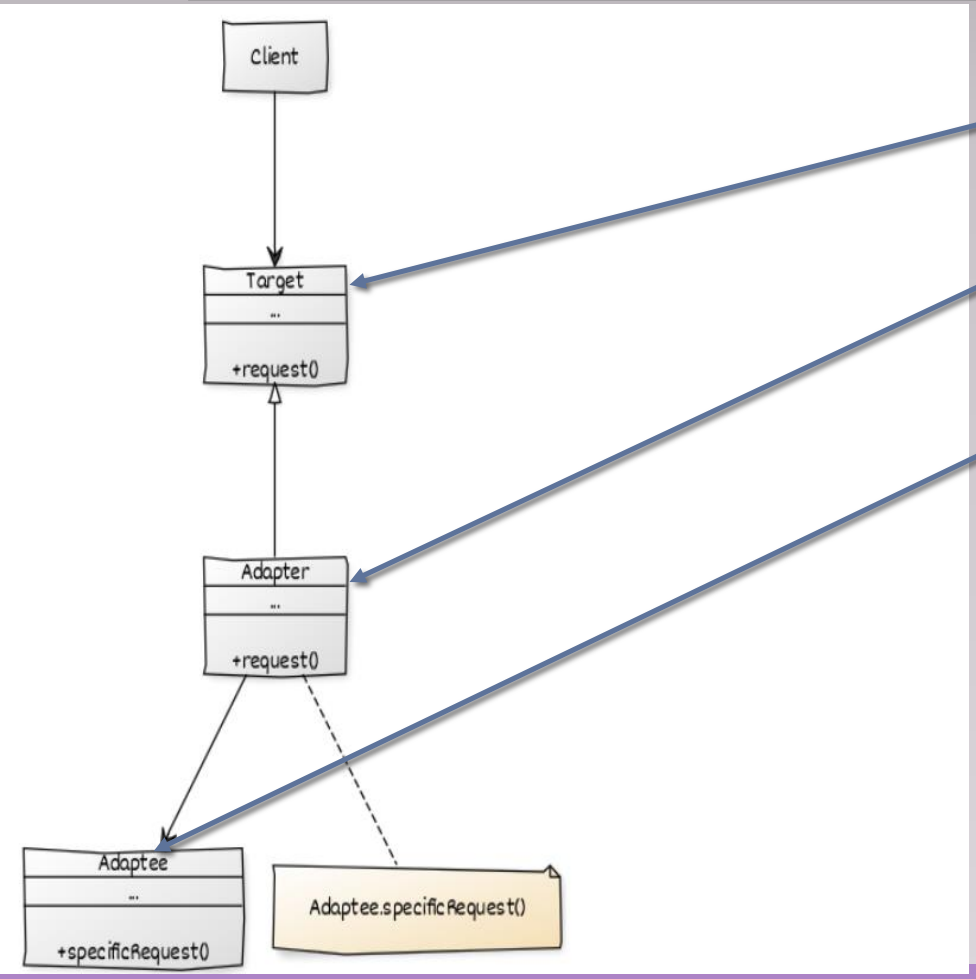
Dr. Öğr. Üyesi Fatih ÖZYURT

Fırat Üniversitesi Yazılım Mühendisliği Bölümü

Adaptör (Adapter Pattern)

- Adaptör tasarım şablonu yardımı ile sistemde mevcut bulunan bir sınıfın sunduğu arayüz (metotları) başka bir sınıf tarafından kullanılabilir şekilde değiştirilir (adapte edilebilir).
- Bu adaptör yardımı ile birbiriyle beraber çalışamayacak durumda olan sınıflar, birlikte çalışabilir hale getirilirler. Yani Bir sınıfın arayüzünü istemcinin beklediği arayüze çevirmeye yarar.
- Bu tasarım şablonu **structural** (yapısal) bir tasarım modelidir.
- Farklı sistemleri kendi sistemimize dahil ederken bazen yapılar uyuşmayabiliyor. Buradaki yapılar arayüzler olabilir. Bu desenin amacı arayüz için bir adapter oluşturarak sanki kendi sistemimize aitmiş gibi çalıştırmayı sağlar. Özetle; **uyumsuz bir yapıyı, istemcinin beklediği bir yapıya getirir**.

Adaptör (Adapter Pattern)



Target: Sistemde kullanılan yapı. (Abstract, Interface vs.)

Adapter: Mevcut sisteme uygulama işlemini yapan sınıf

Adaptee: Mevcut sisteme uygulanmak istenen sınıf.

Adaptör (Adapter Pattern)

- Geliştirilen uygulamada JSON işlemleri için sistemimizin `IJsonSerializer` arayüzünü kullandığını düşünelim.
- İlerleyen süreçte bu işlemleri gerçekleştirmek için 3. parti bir yazılım kullanmak istediniz fakat daha sonradan eklediğiniz yapı sisteminiz ile uyumsuz çünkü sistem `IJsonSerializer` arayüzünü uyguluyor fakat yeni gelen yapı bunu uygulamamaktadır.
- Yeni gelen yapıya da müdahalede bulunamıyorsunuz. Bu sorunu adapter ile aşağıdaki gibi çözüme götürebiliriz.

Adaptör (Adapter Pattern)

```
interface IJsonSerializer
{
    public string SerializeObject(object obj);
}
```

// UML diyagramındaki Target'a denk gelmektedir.
// Sistem bu arayüzü uygulayarak çalışmaktadır.

```
class CustomSerializer
{
    public string Serialize(object obj)
    {
        // Gelen nesneyi serilize etmek için temel operasyonel işlemler..
        // Daha sonradan da gerekli değerin geri döndürülmesi...
        return "serialized with CustomSerializer";

    }
}
```

/ /Sisteme daha sonradan dahil olan yapı.
// Sistemin kullandığı arayüzü **uygulamamaktadır**.
// UML diyagramındaki Adaptee'ye denk gelmektedir.

Adaptör (Adapter Pattern)

```
class CustomSerializerAdapter : IJsonSerializer
{
    public string SerializeObject(object obj)
    {
        CustomSerializer customSerializer = new CustomSerializer();
        return customSerializer.Serialize(obj);
    }
}
```

/ /Daha sonradan dahil edilen yapının sisteme **adepte edilmesi** işlemi.
// Sistemin kullandığı arayüzü **uygular**.
// Bu sayede IJsonSerializer'i uygulayan bir sınıf örneği istendiğinde
// bu sınıf örneğini vererek mevcut koda dokunmadan sistemin devam etmesi sağlanır.
// UML diyagramındaki Adapter'a denk gelmektedir.

Adaptör (Adapter Pattern)

```
class CustomOperation
{
    private IJsonSerializer _jsonSerializer;

    public CustomOperation(IJsonSerializer jsonSerializer)
    {
        _jsonSerializer = jsonSerializer;
    }

    public string SerializeObject(object obj)
    {
        return _jsonSerializer.SerializeObject(obj);
    }
}

var customOperation = new CustomOperation(new CustomSerializerAdapter());

string serializedObject = customOperation.SerializeObject(new object());

Console.WriteLine(serializedObject); // output: serialized with CustomSerializer
```

Adaptör (Adapter Pattern) tasarım şablonu ne zaman kullanılır?

- Sistemde mevcut sınıflar kullanmak istenildiğinde, lakin sınıfın sunmuş olduğu arayüz (metot ve değişkenler) istenilen cinsten değilse.
- Tekrar kullanılabilir ve sistemin diğer bölümlerinden bağımsız bir sınıf ya da komponent oluşturulmak istendiğinde.

Referanslar

1. Özcan acar Design pattern kitabı

2. https://javabeginnerstutorial.com/

3. Yusuf Yılmaz Ders notları

Sorularınız