**Mediator (Arabulucu) Kalıbı**

• **Intent**: Define an object that encapsulates how a set of objects interact. Mediator promotes loose coupling by keeping objects from referring to each other explicitly, and it lets you vary their interaction independently.

• **Amaç**: Bir grup nesnenin birbirleriyle nasıl iletişimde bulunacaklarını soyutlayan bir nesne tanımla. Mediator (arabulucu) nesneleri birbirlerine doğrudan ulaşmaktan alıkoyarak gevşek bağımlılığa destek verir ve aralarındaki iletişimi serbestçe değiştirmenize izin verir.

**PROBLEMS**

**• Nesne-merkezli tasarımda sorumluluklar nesnelere dağıtılır ve nesnelerin birbirleriyle koordinasyon içinde sistemi harekete geçirir.**

• Bunu yaparken nesnelerin birlikteliği (**cohesion**) ve tekrar kullanımı (**reusability**) arttırmaka amaçlanır ama kaçınılmaz olarak nesneler arasındaki bağımlıkları (**coupling**) da artar.

• Çünkü çok sayıda nesnenin birbirleriyle haberleşmesi gerekir.

• En kötü durumda her nesne diğer tüm nesneleri bilir.

• Böyle durumlarda her nesnenin diğer tüm nesneleri bilmesi ve onlarla haberleşmesi bu problemi çözmenin çok karmaşık bir yoludur.

• Bu tür çözümlerde sistemde değişiklik yapmanın maliyeti de yüksek olur.

• Davranış ve iletişimde değişiklikler, her haberleşen nesnenin pek çok alt sınıfını oluşturmayı gerektirir.

• Bu şekilde kullanılan nesnelerin tekrar kullanımı da düşüktür çünkü nesnelerin birbirlerine olan bağımlılıkları yüksektir.

**• GoF, GUI sistemlerinde olan bir diyalog kutusunu (dialog box) örnek verir.**

• Diyalog kutusunda menü, text alanı, button vb. pek çok farklı GUI nesnesi, tabi olarak aralarında pek çok bağımlılıkla birlikte bulunur.

• Örneğin doldurulması zorunlu alanlar boş iken gönderme buttonu aktif, tıklanabilir değildir.

• Farklı GUI nesneleri, diğer pek çok GUI nesnesine bağımlıdır.



• Bu türden, nesneler arasında yoğun bağımlılık olan sistemlerde bağımılıkları bire bir (1-1, one-to-one) haberleşmeyle çözmek ciddi güçlükler çıkarır.

• Sayı olarak çok fazla iletişim kanalı olur,

• Bu iletişim kanalları arasında koordinasyon ve tutarlılık asplamak güçleşir vs.

**SOLUTION**

• Bu şekilde pek çok nesnenin karmaşık yollarla haberleşmesi için bu nesnelerin arasına, hepsini bilen ve koordinasyonu sağlayan bir **Mediator** yani arabulucu nesne koyulabilir.

• **Mediator’ın** amacı nesneler arasındaki iletişimi kontrol ve koordine etmektir.



• Nesneler, arabulucu üzerinden bilgi alış-verişinde bulunur.

• Arabulucu nesne ise diğer nesnelerle nasıl ve hangi durumlarda haberleşeceğini bilir.

• Bu şekilde n tane nesne arasında muhtemel n\*(n-1)/2 tane iletişim kanalı yerine arabulucu nesnenin n tane nesne ile n tane iletişim kanalına sahip olması sağlanır.

• Dolayısıyla **Mediator** kalıbı karmaşıklığı azaltır

**• Mediator tüm arabulucu nesneler için bir üst sınıftır.**

**• Colleague’in alt tipleri olan ConcreteColleague nesneleri de**

**ConcreteMediator üzerinden birbirleriyle haberleşir.**

****

****

• GoF’un GUI sistemindeki problem de **Widget** nesneleri arasına konacak

bir arabulucu, **FontDialogDirector** nesnesi ile çözülebilir.

• İstemci de arabulucu nesne ile haberleşir.





**• Havaalanlarındaki uçak trafiğini yöneten kontrol kuleleri, araç trafiğini**

**yöneten ışık sistemleri, trafik polisleri vs. hep Mediator kalıbının**

**günlük hayattaki uygulamalarıdır.**

**SONUCLAR**

• Mediator kalıbı ile, bu kalıp uygulanmadığında nesneler arasında kurgulanacak many-to-many bağlantılar, one-to-many ile gerçekleştirilir.

• Aksi halde n-1 tane nesnenin arayüzünü bilecek olan nesneler bu durumda sadece arabulucu nesneyi bilirler.

• **Nesnelere dağılacak haberleşme protokolu bilgisi, merkezi olarak mediator nesnesinde toplanır.**

**• Bu şekilde kontrol merkezileşir.**

• Mediator kalıbı ile nesneler arasında gevşek bağımlılığın kurulması

sağlanır.

• Bu şekilde nesnelerin tekrar kullanım ihtimalleri artar.

• Mediator kalıbı ile, aksi taktirde çok karmaşık olacak iletişim

mekanizması basitleşir.

• Mediator kalıbında karmaşıklık arbulucu nesneye kayar çünkü her türlü iletişim kanalı bilgiğsiğ, ilgili süreçler ve iş kuralları arabulucuda birikir.

• Arabulucu nesnenin karmaşıklığını makul seviyede tutmak amacıyla aynı nesne grubu arasındaki haberleşmeyi yönetmek amacıyla birden fazla arabulucu nesneler kurulabilir.

• Bu durumda arabulucu nesneler birbirleriyle haberleşme ihtiyacı duyabilir.

• Ne zaman çok nesne arasında karmaşık ama iyi tanımlanmış haberleşme ihtiyacı varsa **Mediator** kullanılabilir.

• **Mediator** kalıbı, birden fazla sınıfa dağılmış iletişimin, çok alt sınıf oluşturmadan değiştirilebilmesini sağlar.

• **Mediator** kalıbının en problemli tarafı, iletişimin karmaşıklığını, arabulucunun karmaşıklığına dönüştürmesidir.

• **Mediator** kalıbında arabulucu nesne diğer haberleşen nesnelerden çok daha karmaşık olacaktır.

• Arabulucu nesnenin thread-safe olması gerekebilir.

• Bu da arabulucu nesnenin bakımını güçleştirir.

**KULLANIMLAR**

• **MVC’deki** **Controller** bir **Mediator** mıdır?

• **Controller** temelde **View** ile **Model’i** yalıtır ve aralarındaki iletişimi

yönetir.

• **Controller** bu açıdan bir **Mediator** gibi görünür ama onun adı her

zaman **MVC’nin** **Controller’u** olmuştur.

• Bu tür **Controller** kullanımının **Mediator’dan** en temel farkı, her

use case için ayrı bir **Controller** nesnesinin olmasıdır.

• Web’de MVC yapısında **Controller** temelde **View’dan** gelen istekleri önce doğrular (**validation**) sonra iş mantığından sorumlu yapılara iletir (**delegation**) ve sonrasında oluşan **Model** nesnesine ve/ veya gelen cevaba göre farklı **View** nesnelerinden birini seçer ve akışı ona yönlendirir.

• **Controller** ile **Mediator** arasındaki fark burada da söz konusudur.

• **MVC mimari kalıbını uygulayan Spring MVC, JSF, ASP.NET vb. çatılarda genelde tüm use caseler için Front Controller isimli bir nesne bulunur.**

**• Bu kullanımda Front Controller, evrensel bir Controller’dır.**

**• Front Controller gerekli konfigürasyon bilgisini okur ve View’lardan gelen istekleri karşılayan nesnelere (handlers) akışı yönlendirir (dispatching) , onların ürettiği Model nesneleri ile use casein sonunda görüntülenecek View nesnesi arasındaki koordinasyonu yönetir.**

****

**• JPA**, **EntityFramework** vb. **ORM** çerçevelerinde kullanılan ve entityler

ile ilişkisel veri tabanındaki tablolar arasındaki eşleşmeyi yöneten

**Mapper** nesneleri bir mediator gibi çalışır.

• .NET’te MediatR, istemciler arasındaki mesajlaşmayı merkezi bir şekilde yönetmek amacıyla Mediator kalıbını kullanan bir çerçevedir.

• Message Oriented Middleware (MOM), sistemler arası olay (event) mekanizması kuran bir yapıdır ve IBM MQ ya da Microsoft MSMQ gibi ürünler ve Java’da Java Messager Service (JMS), ve .NET Azure Bus Services gibi arayüzlerle gerçekleştirilir.

• MOM sistemleri birer arabulucu gibi çalışır.

**DİGER KALIPLARLA İLİSKİ**

• **Mediator**, nesnelerle haberleşirken **Observer** kalıbını kullanabilir.

• **Observer** kalıbındaki **ChangeManager** bir arabulucudur.

• **Mediator**, **Façade** kalıbınından farkı, ilkinin iki yönlü iletişim sağlarken ikincisinin tek yönlü haberleşme sağlamasıdır.

• Bu yüzden **Façade** kalıbı alt sistemi soyutlarken **Mediator** karmaşık haberleşmeyi basitleştirir.

**• Mediator, Singleton olabilir.**

**UYGULAMA**

• Bir banka şubesine gelen müşterilerin banka çalışanlarından sırayla

hizmet alabilmeleri için gerekli yapıyı Mediator kalıbını kullanarak

tasarlayın.

• Bankada pek çok çalışan olduğunu ve müşterilerin random bir şekilde

bankaya gelip sıralarını beklediklerini düşünün.

• Merkezi bir sıra yönetim sisteminin, müşterilerin sırasını ve çalışanları

koordine etmesi gereklidir.