**Decorator Kalıbı (Pattern)**

• **Intent**: Attach additional responsibilities to an object dynamically.

Decorators provide a flexible alternative to subclassing for extending

functionality.

• **Amaç**: Bir nesneye ek sorumlulukları dinamik olarak ekle. Dekoratörler fonksiyonelliği genişletmek için mirasa esnek bir alternatif sağlar.

• Bu kalıbın ilk ismi “**Wrapper**”dı sonra **Decorator** olarak değiştirildi.

• Dekoratör, bezeyicidir, farklı dekoratörler nesneyi sırayla bezeyerek ona yeni sorumluluklar/özellikler kazandırır.

**PROBLEM**

• Sıklıkla, yeni sorumluluğu/yetkinliği tüm sınıfa değil, nesnelere eklemek isteriz.

• Sorumlulukları sınıfa eklemek için tipik olarak mirası kullanırız.

• Sınıflara yetkinlik eklemek için miras kullanılmasının en temel problemi, **mirasın bir derleme zamanı (compile-time) yapısı olmasıdır.**

• Her yeni yetkinlik için kod değişikliği ve tekrar derleme (recompilation) gerekir.

• Ayrıca böyle çok yetkinliğin ve kombinasyonlarının söz konusu olduğu hallerde, hem derin ve geniş nesne hiyerarşileri oluşur hem de çoklu miras (multiple inheritance) kullanmak zorunda kalırız.

• **n** farklı sorumluluk için kombinasyonlarla **n!** alt sınıfa ihtiyaç olabilir.

**GoF**, bir metin gösteren, **TextView** sınıfını örnek verir.

• Bu sınıfı **ScrollBar’lı** ya da **Border’lı** hale nasıl getirebiliriz?

• Soru, “sınıf mı yoksa nesneler mi bu yeni özelliklere sahip olmalıdır?”dır.



**ÖNEMLİ NOT: Bir sınıfa yetkinlik kazandırmanın tek yolu sınıfa extend edilmek değil, nesneye de yetkinlik kazandırabiliriz. Bu pattern’daki kritik nokta bu.**

**SONUCLAR**

• **Decorator** kalıbı, sınıf mirasından daha esnek bir yapı sağlar.

• Çünkü bir nesnenin sahip olabileceği özellikleri – sorumlulukları önceden öngörerek sınıfa eklemek yerine, çalışma zamanında, isteğe bağlı olarak nesneye eklenmesine izin verir.

• **Decorator** kalıbı, bu şekilde son derece şık ve esnek bir dekorasyon yöntemi sağlar.

• En negatif tarafı ise pek çok ufak **Decorator** nesnesine sahip olmasıdır

**• Dolayısıyla Decoratör kalıbını kullanmadaki amaçlar şunlardır:**

• Diğer nesneleri etkilemeden, nesnelere dinamik olarak yeni

sorumluluklar eklemek,

• Sorumlulukları nesnelerden kaldırmak,

**• Ve özellikle de bunları yapmak için mirasın pratik bir yöntem olmadığı**

**hallerde.**

**KULLANIMLAR**

• Java’da **java.io** paketinde ve **.NET’te** **System.IO.Stream** namespacesinde **Decorator** kalıbı kullanılmıştır.



• Web yapılarındaki filtreler de dekoratör kalıbını kullanırlar.

• Web uygulamalarında istemciden gelen isteğin ya da ona döndürülen cevabın öcne farklı konularda işlenmesine ihtiytaç duyulur.

• Örneğin, oluşturulan bir cevap, önce XML’e çevrilir, sonra XSLT transformasyonuyla bezenir, sonra ziplenir en son olarak da şifrelenir.

• Bu işlemlerin tam tersi gelen istekler için de yapılır.

• Bu yapı dekoratörler ile kurulabilir.

DİĞER KALIPLARLA İLİSKİ

• **Decorator** kalıbının **Composite**‘den farkı, ilkinde sorumluluğun

**Decorator** nesnelerinde iken ikincisinde **Composite** nesnesinde

olmasıdır.

• **Composite’de**, bileşik nesneye eklenen nesneleri yönetmek bileşik

nesnenin sorumluluğundayken **Decorator** kalıbında sorumluluk

dekoratör nesnelerindedir.

• O yüzden **Decorator** kalıbında bütün - parça ilişkisi yoktur, nesnenin

süreç içinde bezenmesi söz konusudur.

• **Decorator**, **Adapter**‘den de farklıdır çünkü **Decorator** nesnenin arayüzünü değiştirmez, sadece sorumluluk ekler.

• **Adapter** ise uyumsuz nesneye tamamen yeni bir arayüz sağlar.

**UYGULAMA**

1. Aşağıdaki problemi Decorator kalıbıyla modelleyip kodlayın:

• Bir iş yerinde maaşlar, temel maaşın üzerine, çalışanların sahip oldukları

yetkinliklere göre eklenen tazminatlarla hesaplanmaktadır.

• Var olan tazminatlar ise şunlardır:

• Yabancı dil,

• Engellilik,

• Master ve Ph.D. sahibi olma,

• Sertifika sahibi olma,

• Yurt dışı ofisinde çalışma.

2. Aşağıdaki problemi **Decorator** kalıbıyla modelleyip kodlayın:

• Bir araç tam koruma (kasko) poliçesinin fiyatı aşağıdaki etkenlerle

belirlkenmekte, etkenler fiyatı arttıran ya da eksilten katkıda

bulunmaktadır:

• Aracın yaşı,

• Kullananın yaşı,

• Kullananın kaza geçmişi,

• Ana kullanım yerleşim merkezi (İstanbul gibi metrokent, Ayvalık gibi

kasaba ya da köy)

• Araçtaki donanım:

• Çelik jantlar

• Değerli elektronik ses vb. sistemler