**Design Pattern’ler**

**Patternler 3 sınıfa ayrılır:**

**-**Creational Patterns (Yaratıcı Desenler): Bu patternler nesnelerin oluşma şeklini yönetir. Amacı nesne oluşumunu daha esnek hale getirmektir.

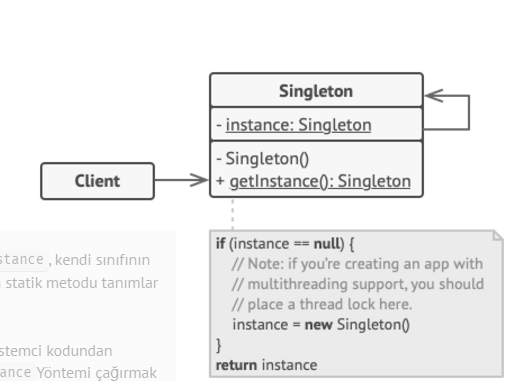
- Structural Patterns (Yapısal Desenler): Nesnelerin ve yapıların bir araya gelerek büyük yapılar oluşturmasına yardımcı olur.

-Behavioral Patterns (Davranışsal Desenler): Nesneler arası iletişimi yönetir.

Patternler’in ana başlıkları bu şekildedir.

**Creational Patterns**

**Singleton:** Singleton bir sınıfın yalnızca bir örneğinin olup o sınıfı public olarak tüm sınıflarda kullanmamızdır.



**Factory Method:** Bu pattern üst sınıfta nesne oluşturmak için arayüz sağlayan ancak nesne oluşturma işini alt sınıflara bırakan patterndir.

metin, taslak, diyagram, tasarım içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Abstract Factory:** Bu pattern birbiriyle ilişkili veya bağımlı nesneleri somut sınıflarını belirtmeden oluşturmak içi kullanılır. Factory methoda benziyor ancak farkı birden çok ürün ailesiyle bağlanabilmesi.

metin, taslak, yazı tipi, ekran görüntüsü içeren bir resim

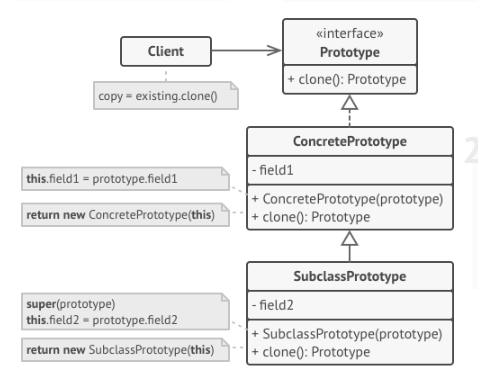
Açıklama otomatik olarak oluşturuldu

**Builder:** Bu pattern karmaşık olan nesneleri adım adım oluşturmamızı sağlıyor. Nesne oluşturulurken her bir parçası birer adımda belirlenir ve en sonunda final bir nesne oluşur.

metin, taslak, ekran görüntüsü, dikdörtgen içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Prototype:** Bu pattern nesneleri kopyalamamıza yarar. Bu pattern sayesinde büyük ve karmaşık nesneleri tekrar tekrar oluşturmak yerine klonunu oluşturabiliyoruz.



**Structural Patterns**

**Adapter:** Bu pattern birbirinden uyumsuz 2 arayüzün birlikte çalışmasını sağlar.

metin, ekran görüntüsü, yazı tipi, makbuz içeren bir resim

Açıklama otomatik olarak oluşturuldu

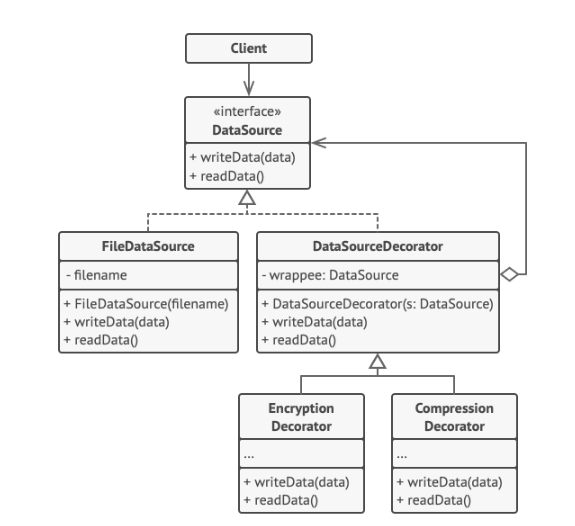
**Bridge:** Bu pattern büyük sınıfları veya yakın ilişkili sınıfları birbirinden ayrı bir şekilde geliştirmemize olanak sağlar.

metin, ekran görüntüsü, diyagram, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Composite:** Bu pattern nesneleri ağaç yapılarına ayrı nesnelermiş gibi işlem yapmamızı sağlar. Bu pattern ağaç benzeri yapıları geliştirmek için idealdir. metin, ekran görüntüsü, makbuz, diyagram içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Decorator:** Bu pattern nesnelere yeni özellikler eklememizi sağlar. 

**Facade:** Bu pattern büyük ve karmaşık olan bir sistemin alt parçalarını gizleyerek kodu daha sade hale getirir.

metin, diyagram, ekran görüntüsü, taslak içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Fly Weight:** bu pattern verimlilik için kullanılır. Birden fazla nesne arasında ortak bölüümleri paylaşır. Mevcut belleğe daha çok nesne sığdırmamıza yarar.

metin, ekran görüntüsü, diyagram, makbuz içeren bir resim

Açıklama otomatik olarak oluşturuldu

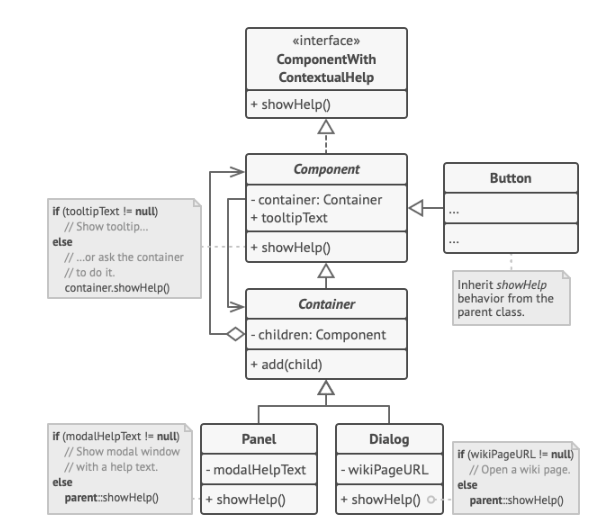
**Proxy:** Bu pattern, bir nesnenin yerine geçici olarak bir vekil nesne koyarak gerçek nesneye olan erişimi kontrol etmek için kullanılır.

metin, makbuz, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Behavioral Patterns**

**Chain Of Responsibility:** Bu pattern istekleri bir işleyici zincir boyunca iletmemizi sağlar. Her işleyici bir isteği ya işler ya da bir sonraki işleyiciye aktarır.



**Command:** Bu pattern bir isteği, istek hakkında tüm bilgileri istekten bağımsız bir nesneye dönüştürmemize yarar. Bu pattern geri alma ve yineleme işlemleri için idealdir.

metin, diyagram, plan, paralel içeren bir resim

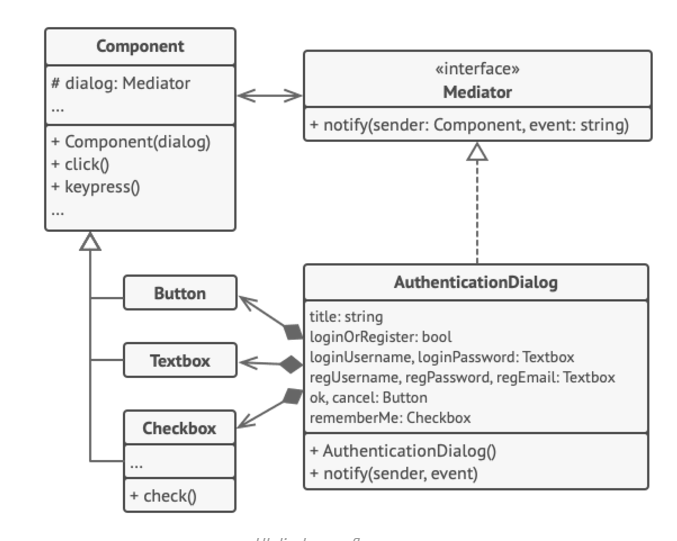
Açıklama otomatik olarak oluşturuldu

**Iterator:** Bir koleksiyonun elemanlarına sırasıyla erişmemize yarar. Koleksiyonun dış elemanlarını gizler.

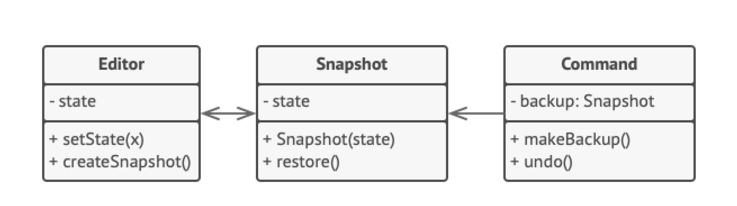
metin, makbuz, diyagram, paralel içeren bir resim

Açıklama otomatik olarak oluşturuldu

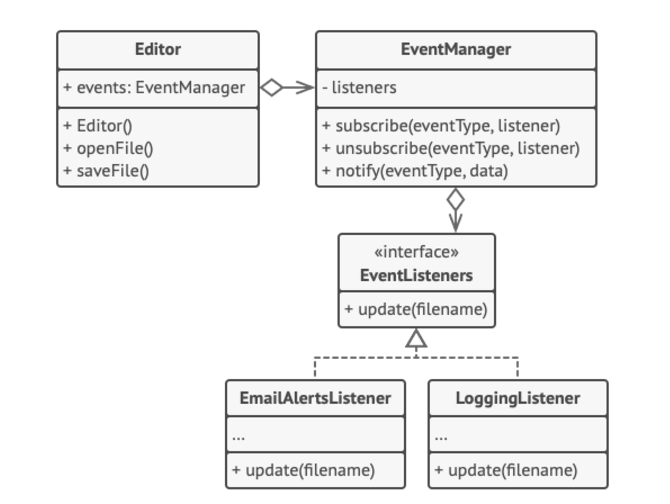
**Mediator:** Bu pattern bir nevi arabuluculuk görevi görür. Nesneler arasındaki karmaşık bağımlılıkları azaltmamıza olanak tanır. Nesneler arasındaki doğrudan iletişimleri kısıtlayarak bir mediatör nesnesi ile iş birliği yapmaya zorlar.



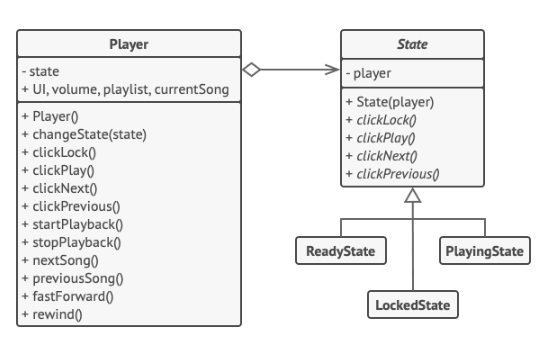
**Memento:** Bu pattern bir nesnenin bilgilerinin ayrıntılarını açığa çıkarmadan önceki durumunu kaydetmemize ve istediğimiz zaman geri almamıza yarar.



**Observer:** Bu pattern nesnelerin gözlemlediği bir nesnenin başına gelen herhangi bir olayda gözlemleyen nesneleri bilgilendiren bir abonelik mekanizması.



**State:** Bu pattern bir nesnenin iç durumu değiştiği zaman davranışının da değişmesine izin veren paterndir.



**Strategy:** Bu pattern bir algoritma ailesi tanımlayıp her birinin ayrı sınıfta bulundurmamıza ve nesnelerin her birini birbiri yerine kullanabilmemizi sağlar.