İleri Java Programlama

Design Patterns

Emir ÖZTÜRK

*İçerik ve şekiller refacoring.guru sitesinden alınmıştır.

Design Patterns

Design Pattern Nedir

- Yazılım geliştirme sürecinde oluşan problemler
- Blueprint
- Kod veya framework değil
- İmplementasyon detayı belirtmez
- Daha önce yaşanmış standart problemlerin çözümü
- Problemlerin belirli bir yolla çözülmesinden sonra isimlendirilmesi

Design Patterns

Design Pattern Faydaları

- Daha önce denenmiş ve test edilmiş çözümler
- OOP dizayna uygun
- Patternlerin bir standart sağlaması
- Herkesin aynı dilde konuşması
- Her yere uygulanmalı mı?
  - Fazla genelleştirme

Design Patterns

Design Pattern Kategorileri

- Creational
  - Esneklik ve yeniden kullanılabilirliğin arttırılması adına nesne oluşturma mekanizmaları
- Structural
  - Nesnelerin ve sınıfların daha büyük yapılara esnekliği kaybetmeden birleştirilmesinin yolları
- Behavioral
  - Nesnelerin sorumluluklarının atanması ve aralarındaki haberleşmesinin tanımlanması

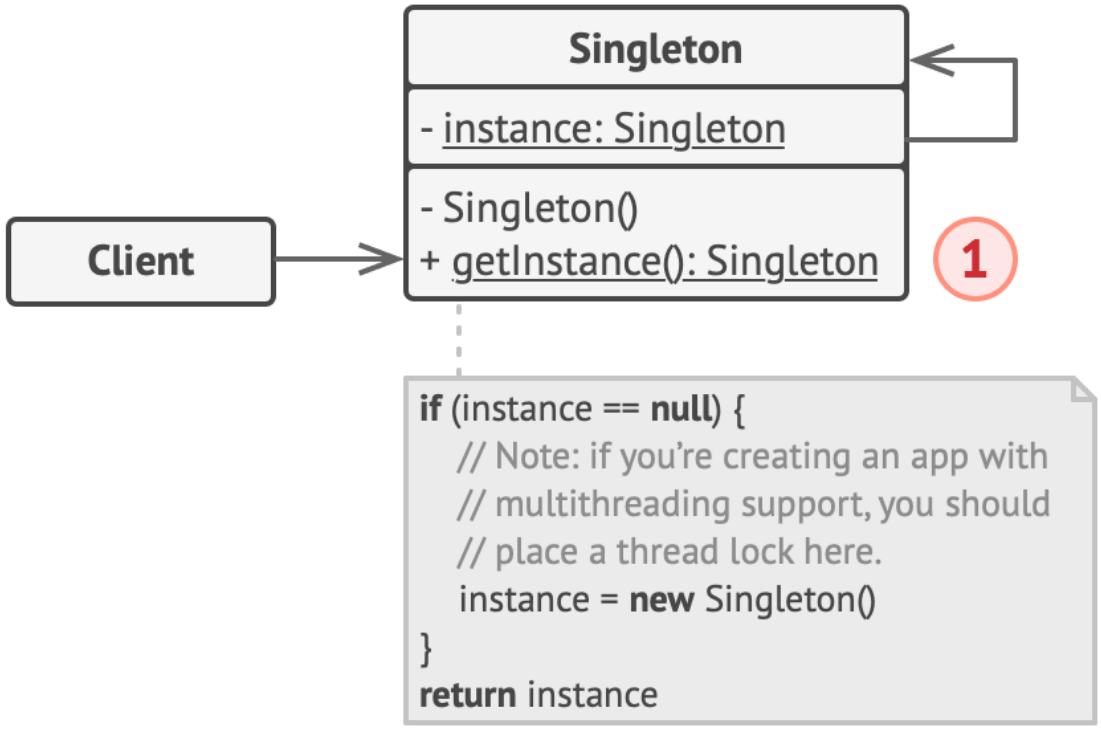
Creational Patterns

Singleton

- Bir sınıfın yalnızca tek örneğe sahip olması
- Her nesne isteyen duruma bu örneğin verilmesi
- Ortak paylaşılan bir kaynağa erişen sınıf için

Creational Patterns

Singleton



Creational Patterns

Singleton

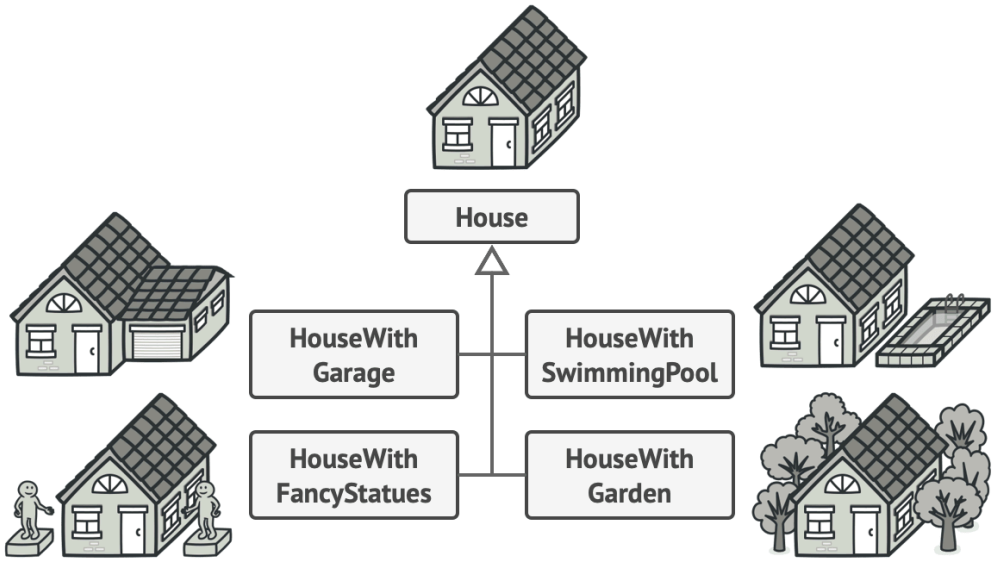
- Sınıfın tek örneği
- Global erişim
- İlk istendiğinde örnek alma
- Performans
- Multithread işlemede problem
- Single responsibility principle'a karşı

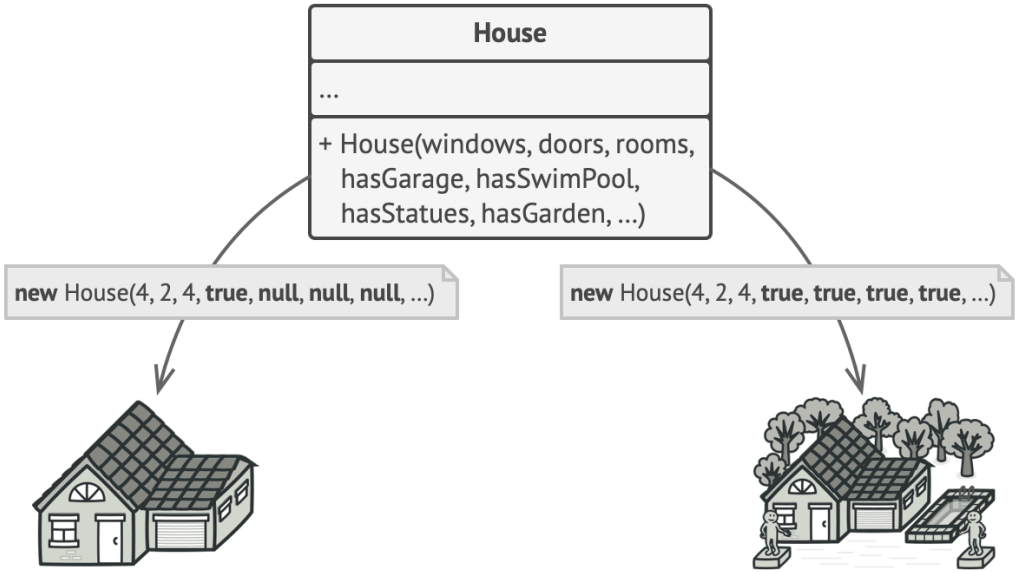
Creational Patterns
Builder

- Karmaşık bir nesnenin oluşturulması
- Bu nesnenin oluşturma adımlarının farklılaşması
- Çok fazla alt nesne
- Tüm oluşturma adımlarının üst nesnede toplanması
- Nesnenin çok büyümesi
- Nesnenin parçalarını oluşturacak builder bir sınıfın yazılması
- Farklı ihtiyaçlara göre builder çağırılması

Creational Patterns

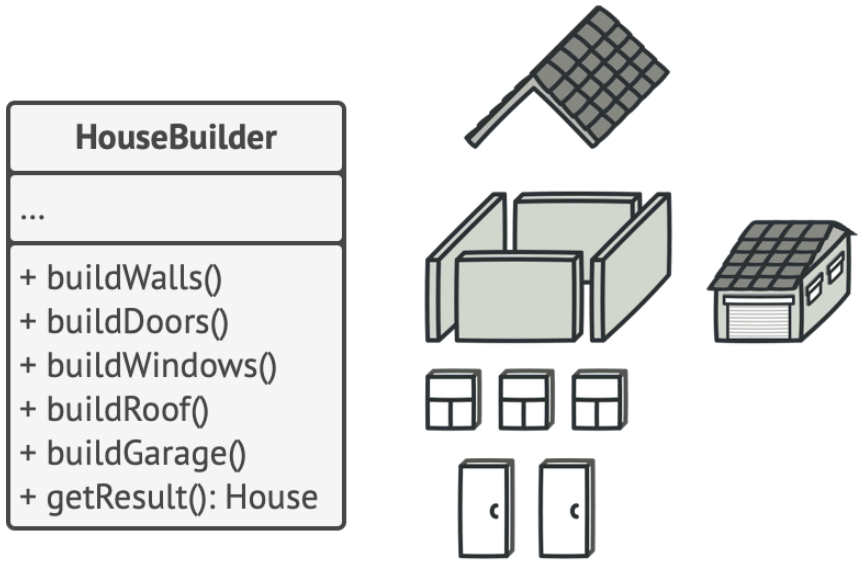
Builder





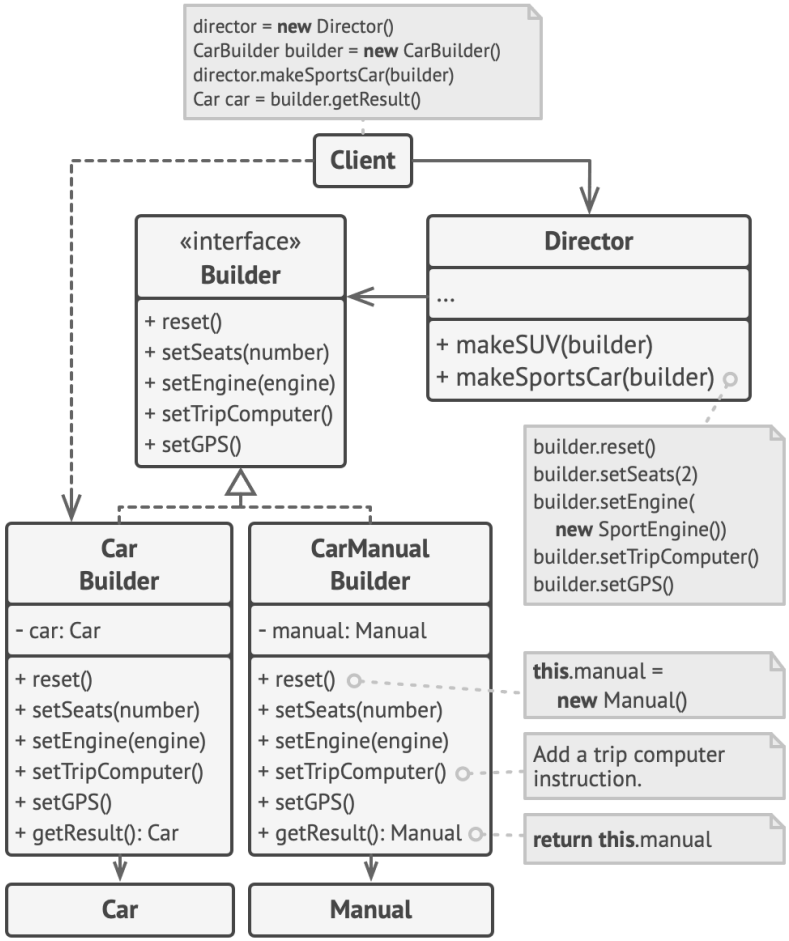
Creational Patterns

Builder



Creational Patterns
Builder

- Farklı builder gruplarını oluşturan metotlar
  - Director
  - Zorunlu değil



Creational Patterns

Builder

- Kodun tekrar kullanılabilmesi
- Parçalara ayırma işlemi sayesinde farklı işlemler için kullanılabilmesi
- Single Responsibility Principle

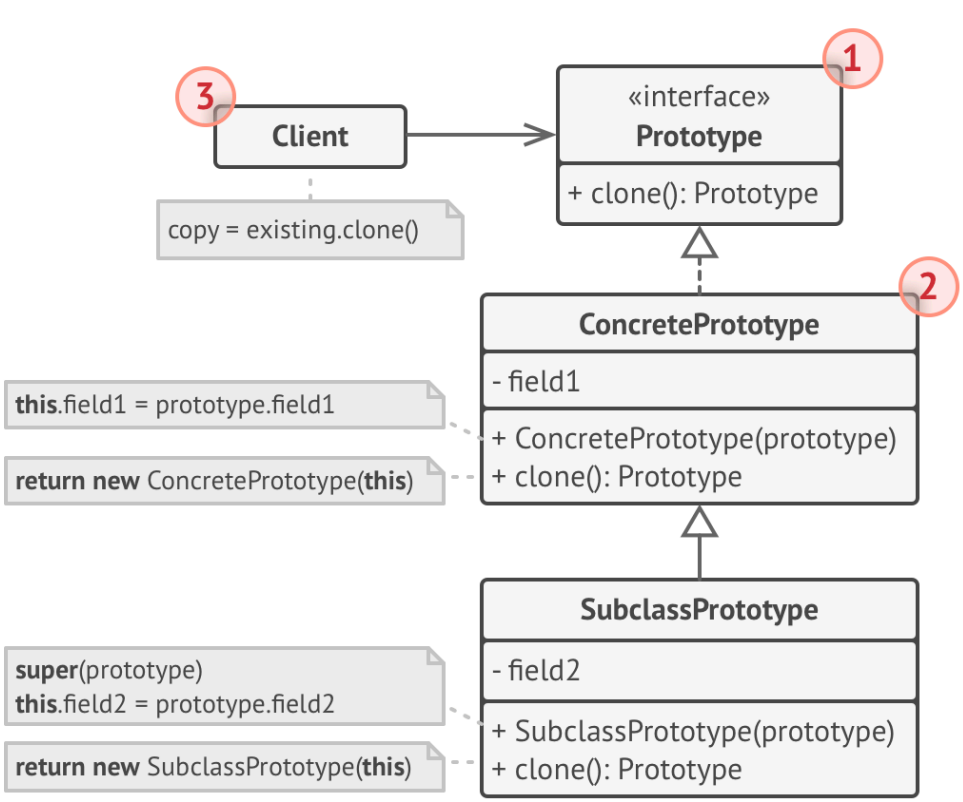
Creational Patterns

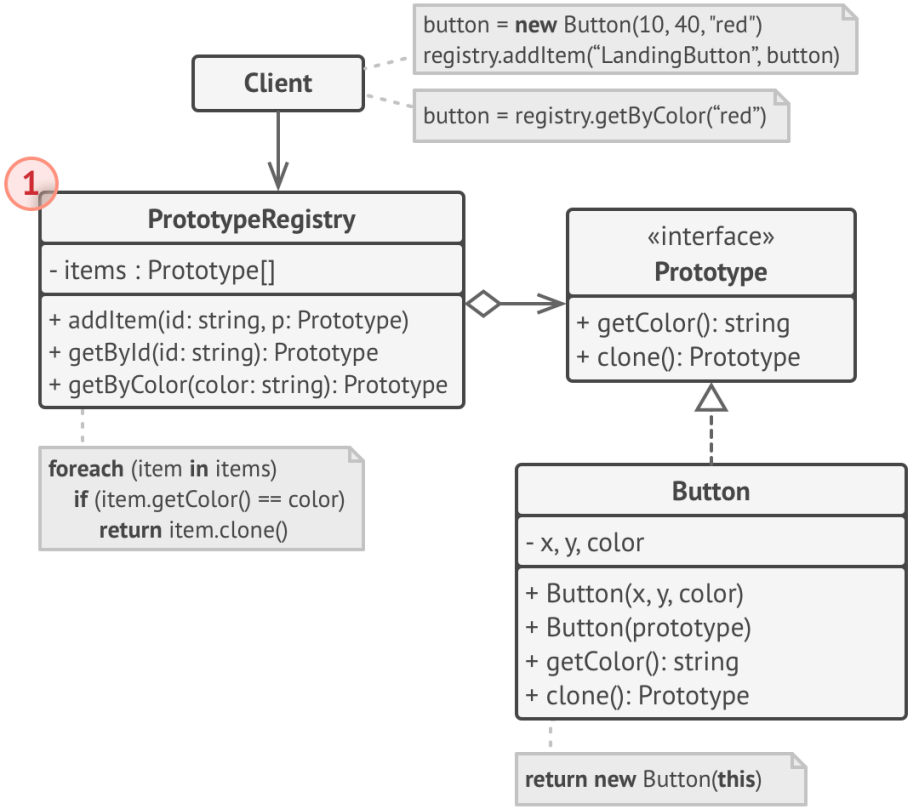
Prototype

- Bir nesnenin kopyalanmak istenmesi
- Nesnenin tüm alanlarının da sırayla kopyalanması
  - Kopyalamada nesnenin erişilemeyecek alanlarının da bulunması
- Nesnenin soyut bir şekilde arayüzden geliyor olma ihtimali
- Nesneye clone() metodunun eklenmesi

Creational Patterns

Prototype





Creational Patterns

Prototype

- Arayüzler üzerinden klonlama desteği
- Karmaşık objelerin kolay kopyalanması
- Tekrar eden nesne alma kodunun ortadan kalkması

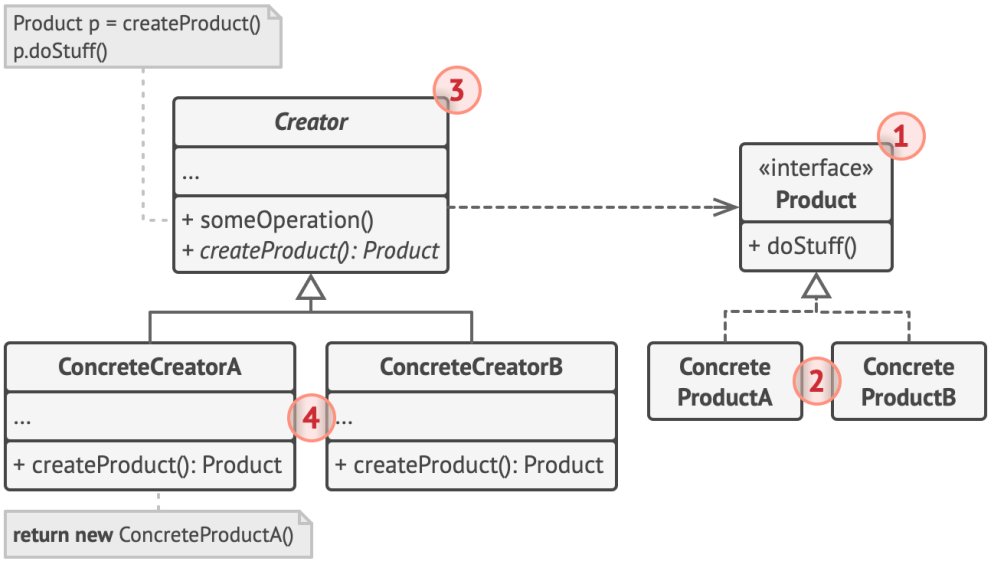
Creational Patterns

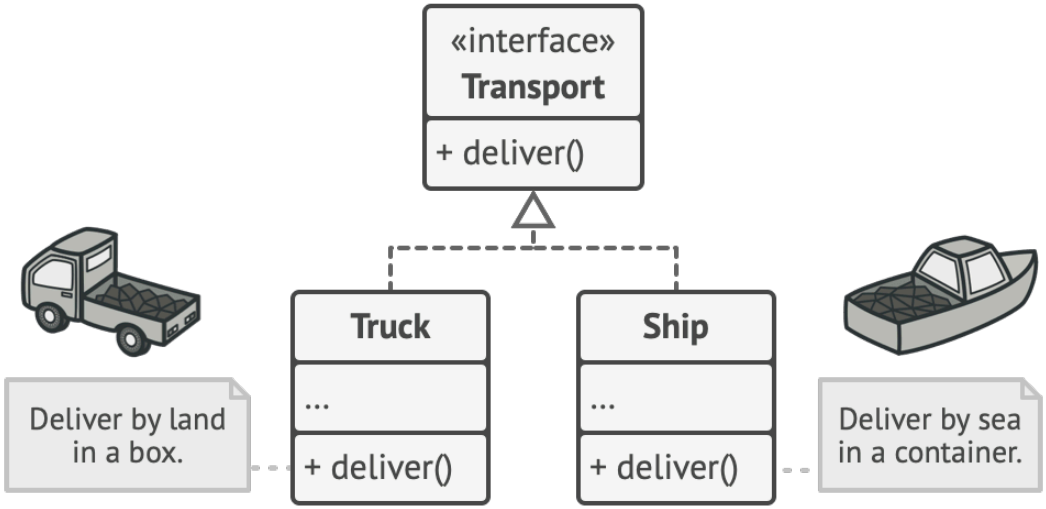
Factory Method

- Bir üst sınıf nesnelerin üretilmesi
  - Alt sınıfların üretilen nesneleri değiştirebilmesi
- Sorun:
  - Bir uygulamanın tanımlanan bir sınıfa bağlı olarak yazılması
  - Yeni tanımlanacak sınıfların uygulamaya dahil edilmesindeki problem
- Çözüm:
  - Sınıfların yapıcı çağırımı yerine bir metot ile çağırılması

Creational Patterns

Factory Method





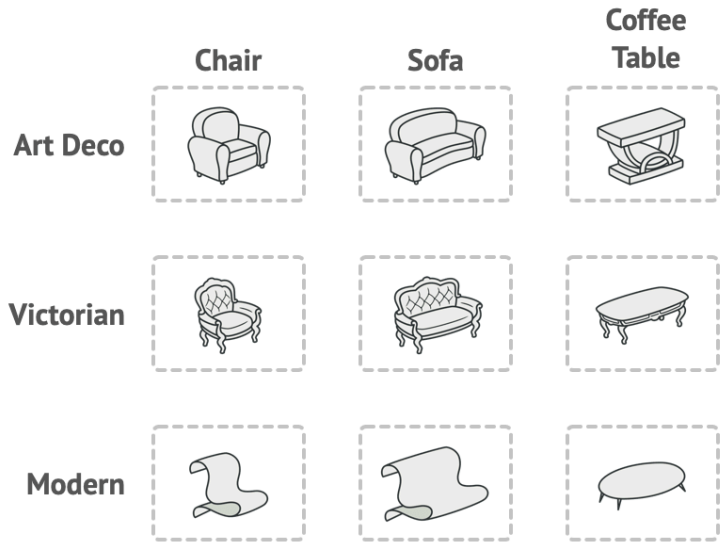
Creational Patterns

Factory Method

- Tek arayüz
  - Standartlaştırma
- Sınıf sayısının belirsizliği
- Pooling
- Üretim kodunun tek noktada toplanması
  - Single responsibility Principle
- Tight coupling kaldırma
- Open / Closed Principle

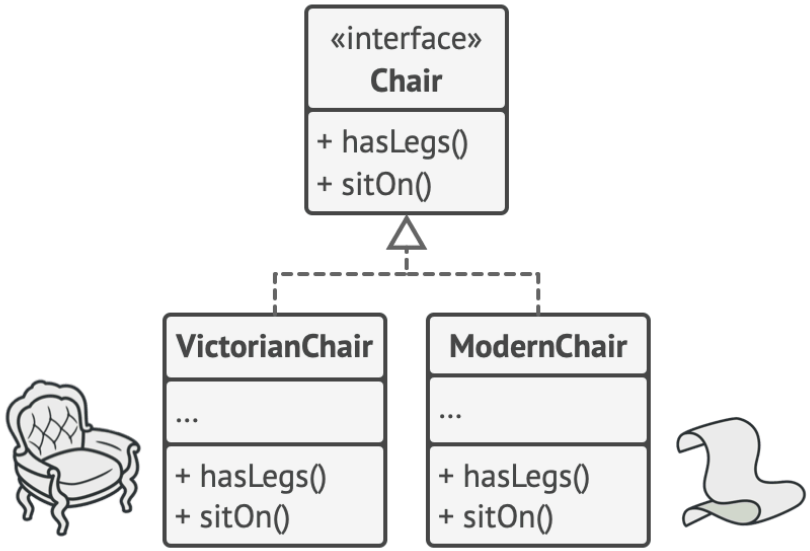
Creational Patterns
Abstract Factory

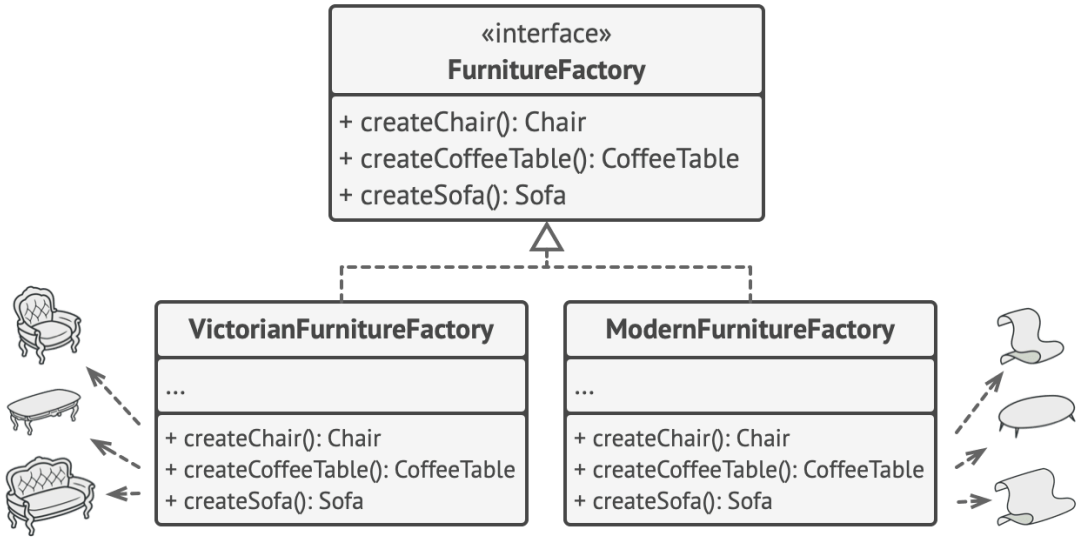
- Factory Method pattern
  - Tek bir arayüz
  - Birden fazla sınıf
- Birden fazla gruba sahip durumlar
- Tüm factory methodların bir abstract üst sınıfı



Creational Patterns

Abstract Factory





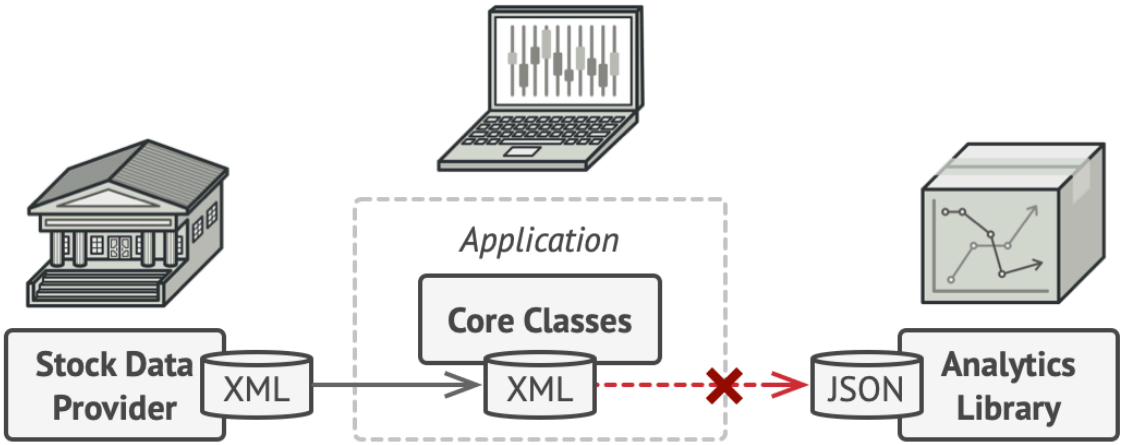
Creational Patterns

Abstract Factory

- Client tarafı
  - Hangi factory sınıfı
  - Hangi türden nesne
- Belirli metotların implement edilmesi
- Üretim kodunun tek noktada toplanması
  - Single responsibility Principle
- Tight coupling kaldırma
- Open / Closed Principle

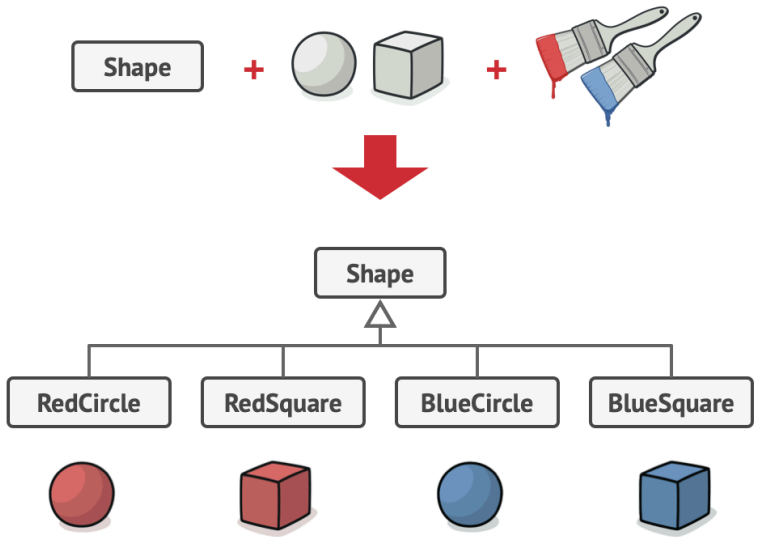
Structural Patterns
Adapter

- Birden fazla formatın bulunduğu uygulamalar
- Formatın tek noktada toplanmak istenmesi
- Uyumluluk
- Single responsibility
- Open Closed



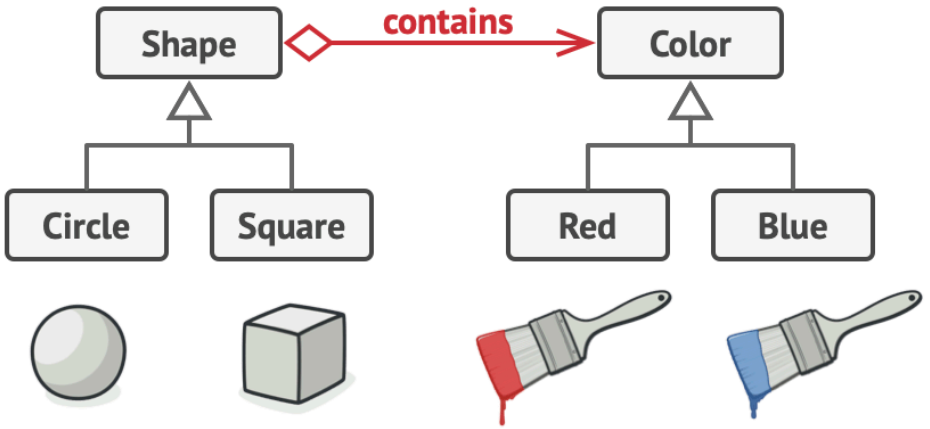
Structural Patterns
Bridge

- Birden fazla gruplandırılabilir özelliğin ayrılması
- Olasılık sayısının azaltılmasının amaçlanması
- Kalıtım yerine hiyerarşi
- Nesne içerisinde başka nesnenin içerilmesi



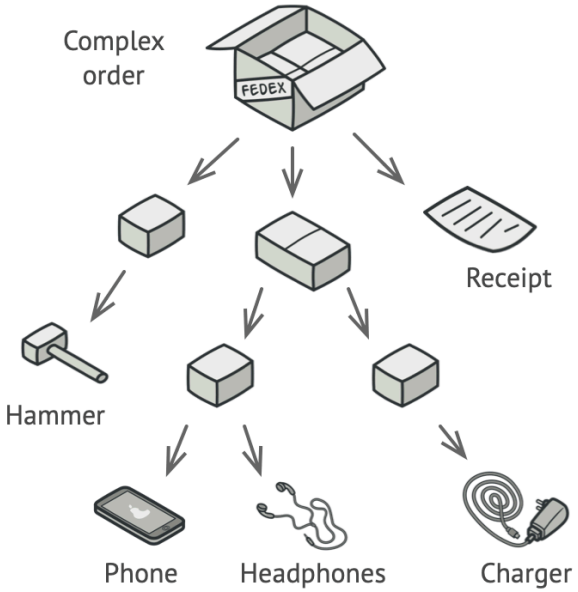
Structural Patterns
Bridge

- Platform bağımsız sınıf tasarlama
- Client tarafı yüksek seviye soyutlanmış sınıflar kullanır
- Open Closed
- Single Responsibility



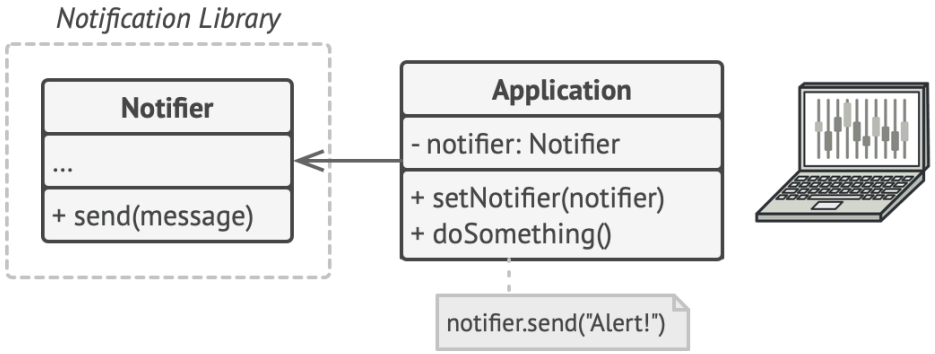
Structural Patterns
Composite

- Nesneleri ağaç yapısına yerleştirip kullanma
- Hesaplanması gereken değişkenlerin hesaplama işinin alt sınıflara verilmesi
- Recursive
- Container'ların olduğu durumda uygun
- Open Closed



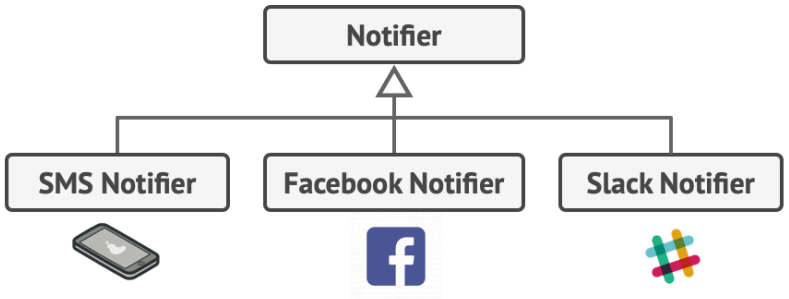
Structural Patterns
Decorator

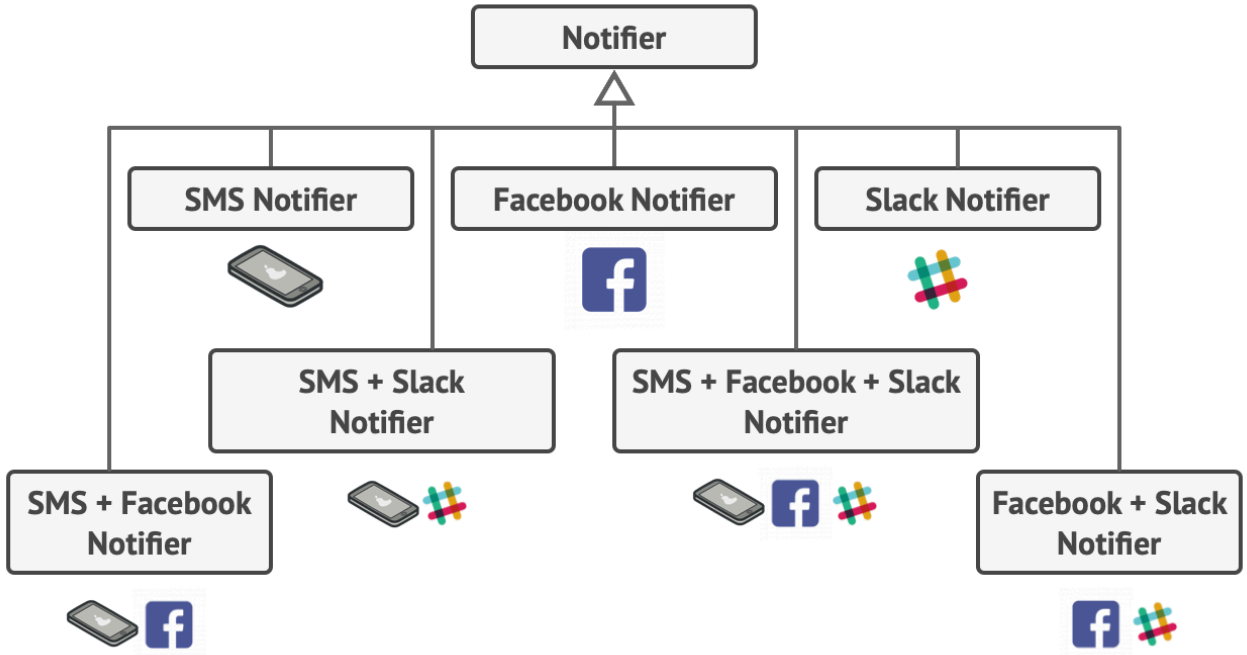
- Bir sınıfın davranışını genişletmek için bu sınıfı başka bir sınıfla sarmalama
- Problem:
  - Sınıfın ihtiyaç duyduğu özelliği eklemek
- Kalıtım
  - Çok fazla alt sınıf
  - Çok fazla olasılık



Structural Patterns

Decorator

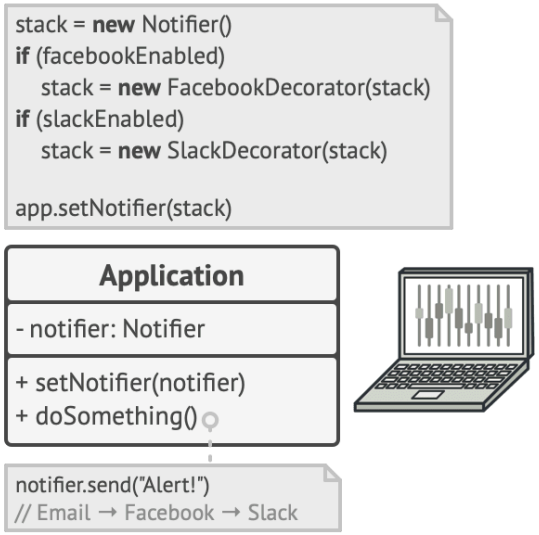


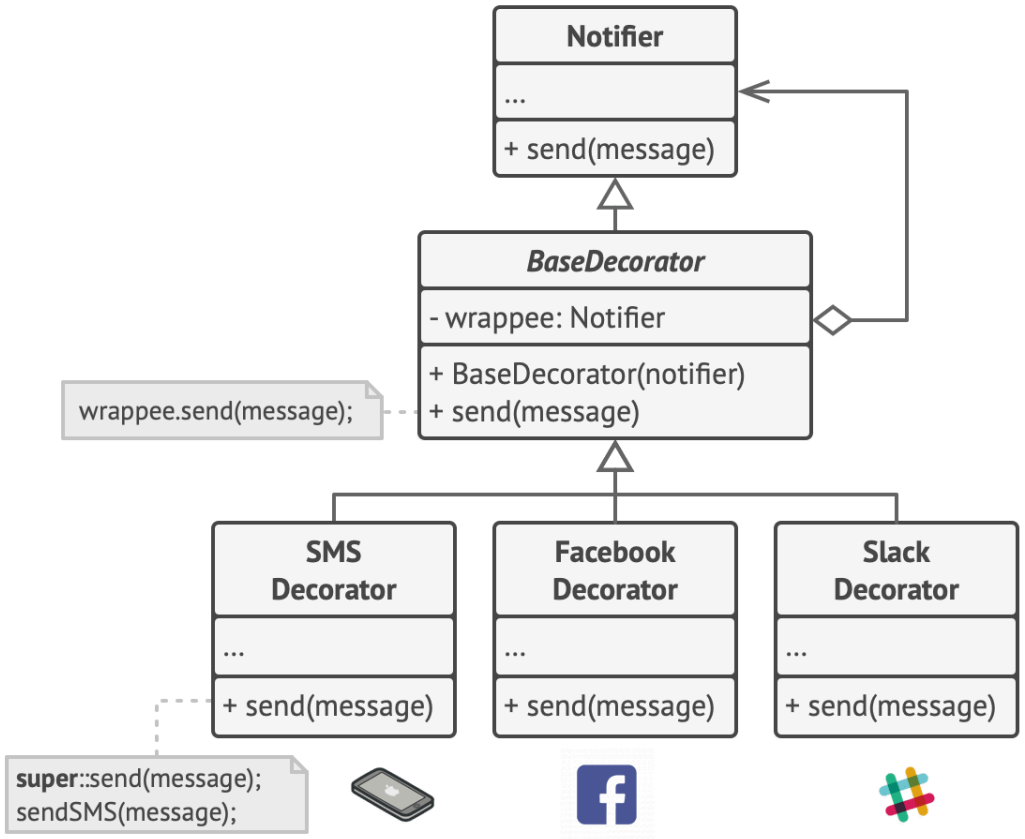


Structural Patterns

Decorator

- Aggregation
- Composition
- Helper / Wrapper





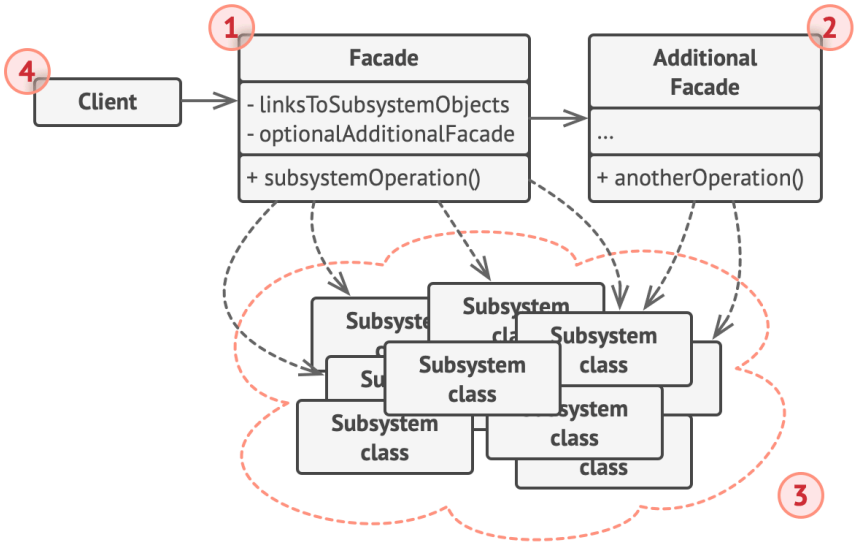
Structural Patterns

Decorator

- Nesne davranışını alt sınıf olmadan genişletmek
- Nesne üzerine runtime’da görev ekleyip çıkartabilmek
- Bir nesneyi birden fazla decorator içerisinde kullanabilmek
- Single Responsibility

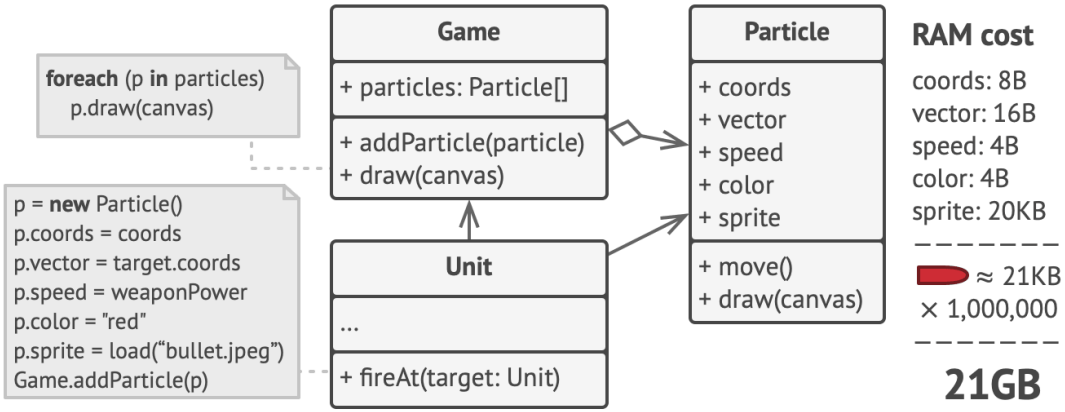
Structural Patterns
Facade

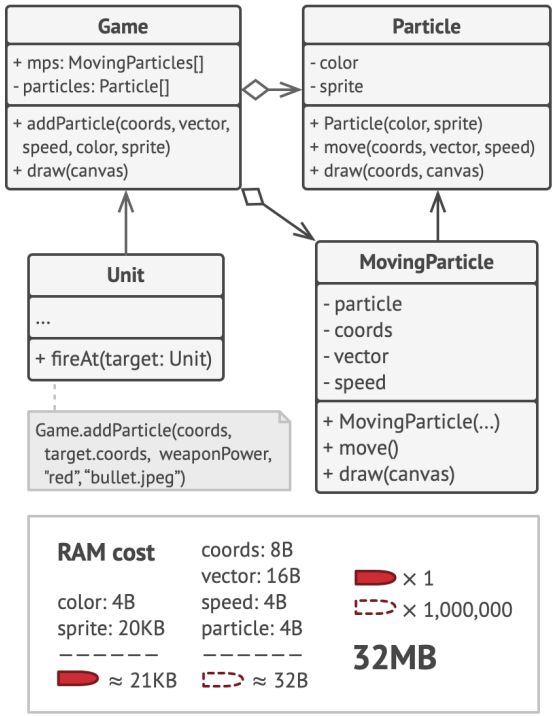
- Karmaşık sınıf setleri, kütüphane veya framework'lere basit bir arayüz sunmak
- Sadece ihtiyaç duyulan özelliklerin client'a sunulması
- İzolasyon
- Soyutlama



Structural Patterns
Flyweight

- Ram ihtiyacı
- Ortak kaynakları kullanan objelerin durumları
- Durum paylaşımı
- Oyun motorları

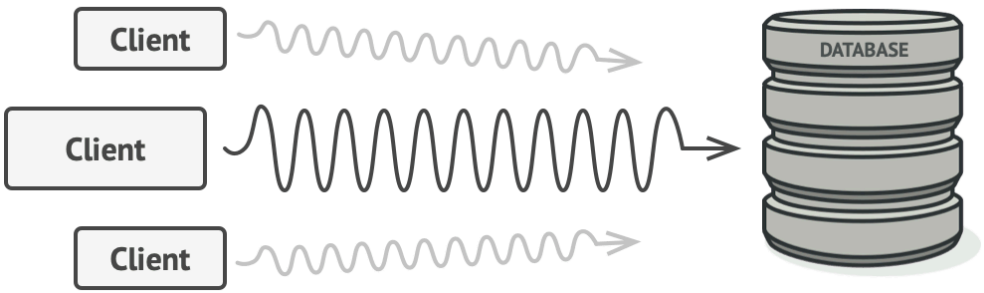


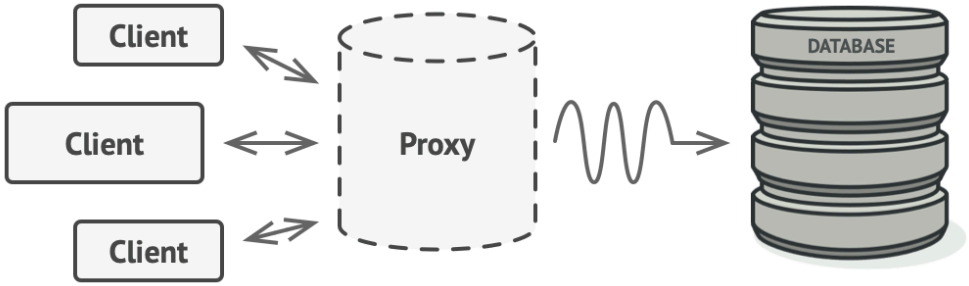


Structural Patterns

Proxy

- Başka bir nesne için yer tutucu
- Çok kaynak tüketen bir nesnenin ikamesi
- Lazy initialization
- Erişim kontrolü
- Caching

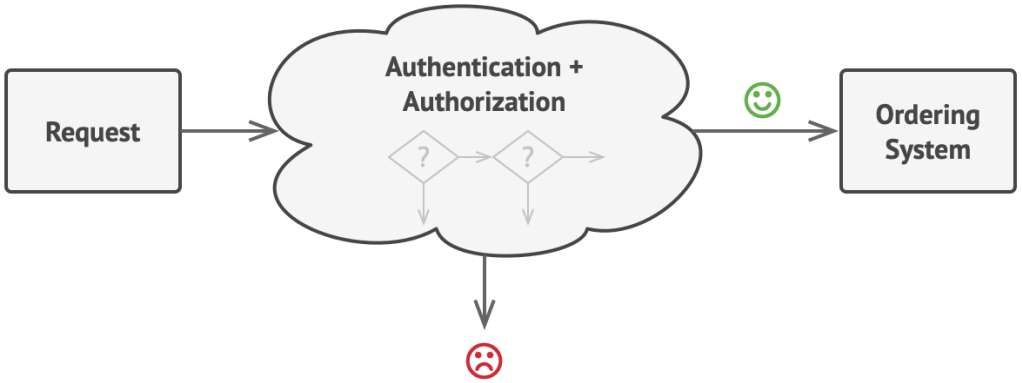


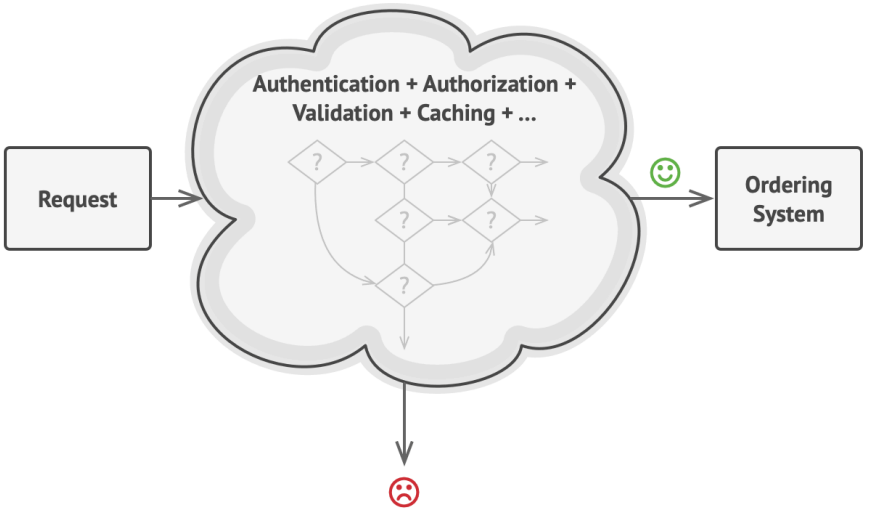


Behavioral Patterns

Chain of Responsibility

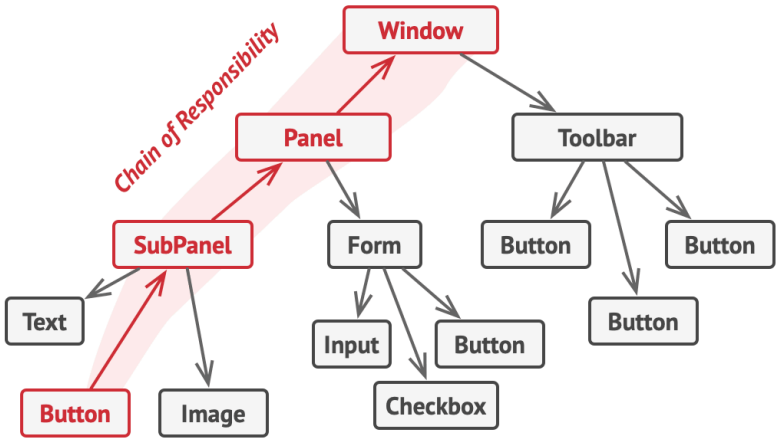
- İstek zinciri
- Zincir boyunca isteklerin ilgili sınıflarca icrası
- Yapılacak görevlerin tek sınıfta toplanmaması

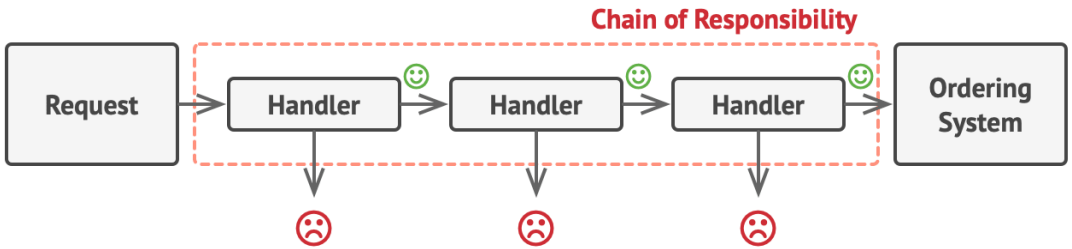




Behavioral Patterns
Chain of Responsibility

- Her sınıfın sırayla kendi ile ilgili kısmı incelemesi
- Onaylanma sonucu bir sonraki sınıfa iletim
- Pencerelerde olay aktarımı

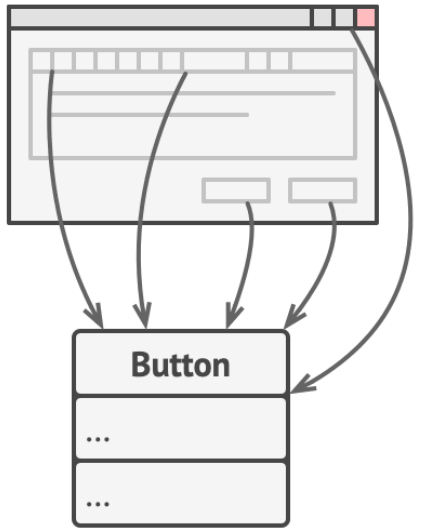


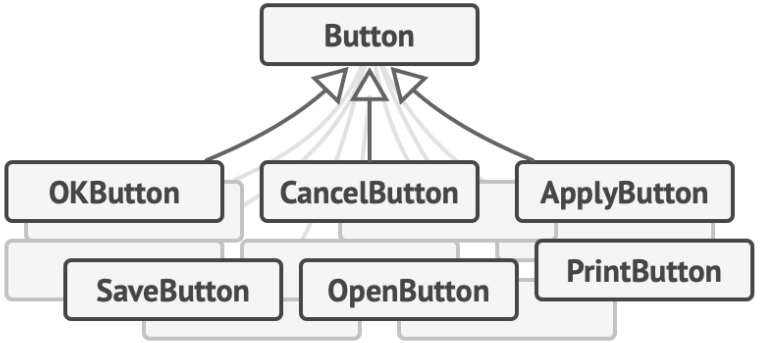


Behavioral Patterns

Command

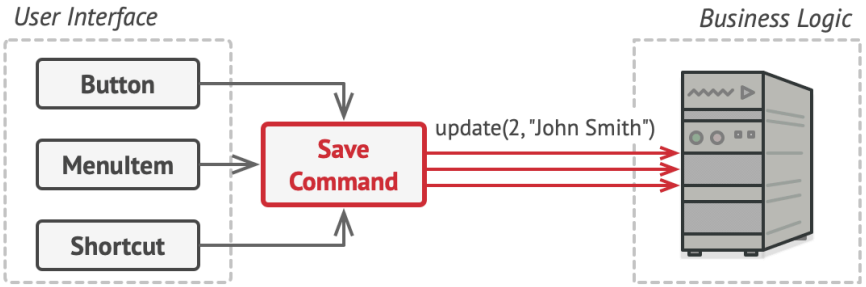
- Tekrar edilen komutların ayrı bir nesne olarak açılması
- Farklı nesnelerin farklı iş yapma gerekliliği
- Kalıtımla çok fazla nesne olması

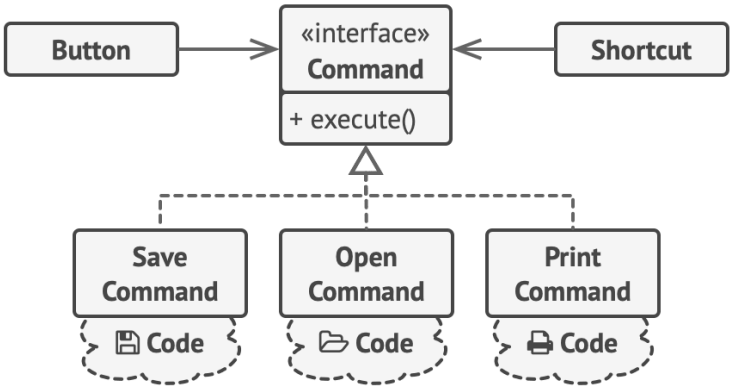




Behavioral Patterns
Command

- Uygulanacak işlemin sınıftan ayrılması
- İşlemlerin tek bir arayüz ile gruplanması
- Sınıfın istenilen işlemi arayüz yolu ile kullanması

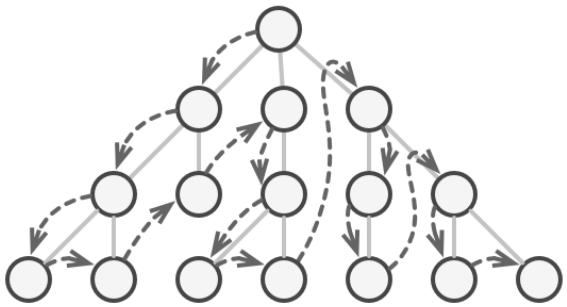


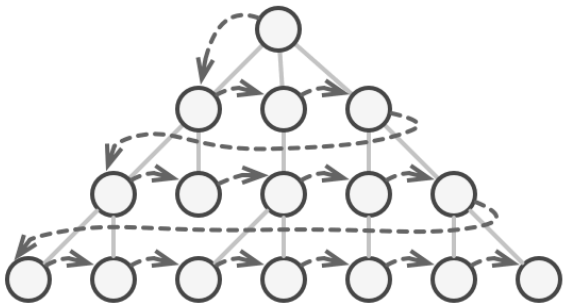


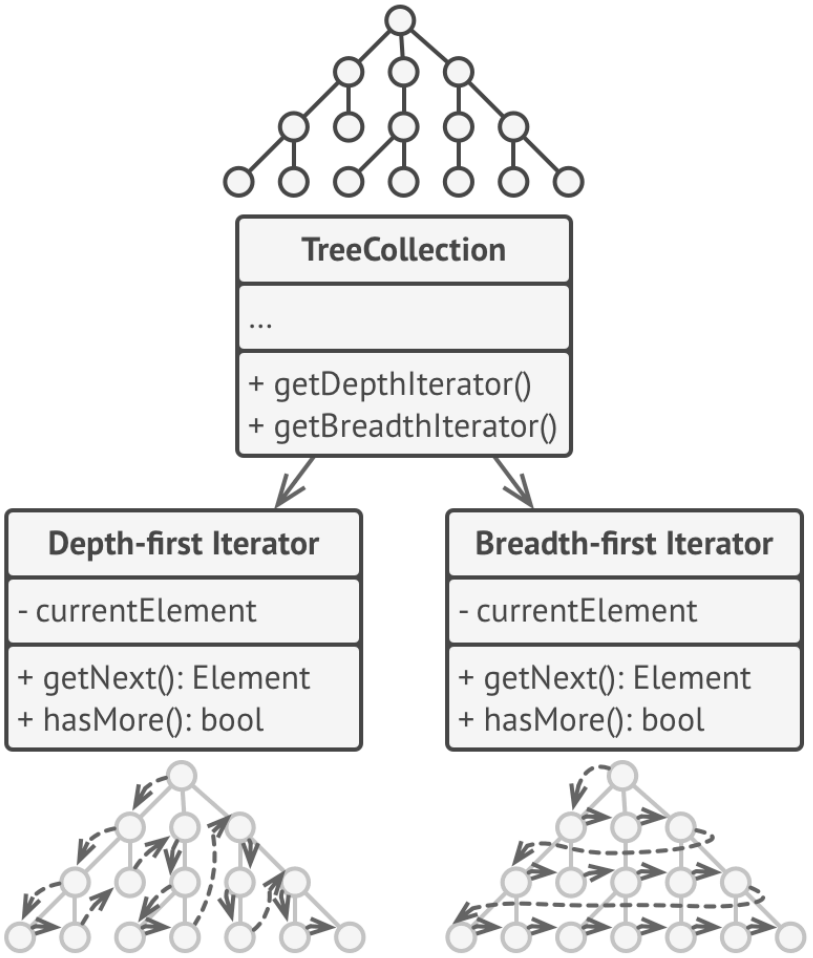
Behavioral Patterns

Iterator

- Dinamik veri yapıları
- Yapıların saklanma karmaşıklığı
- Bu yapılardan veri elde etmek için iterator
- Iterator implementasyonunun gizlenmesi

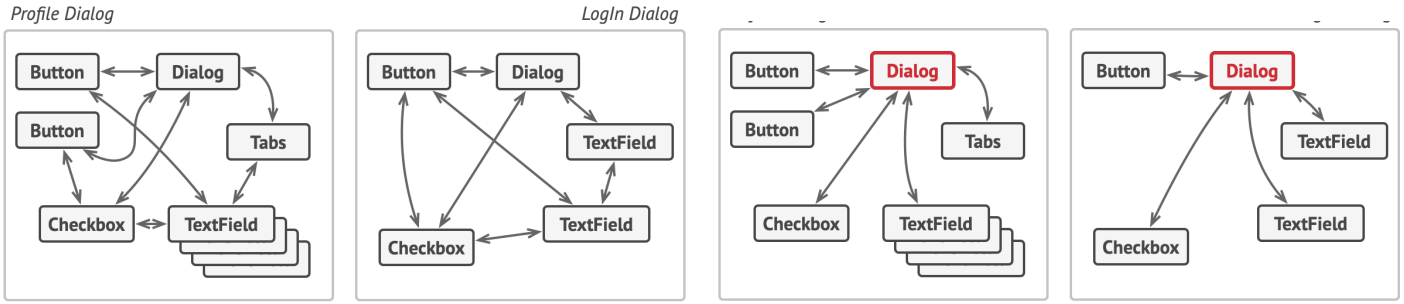


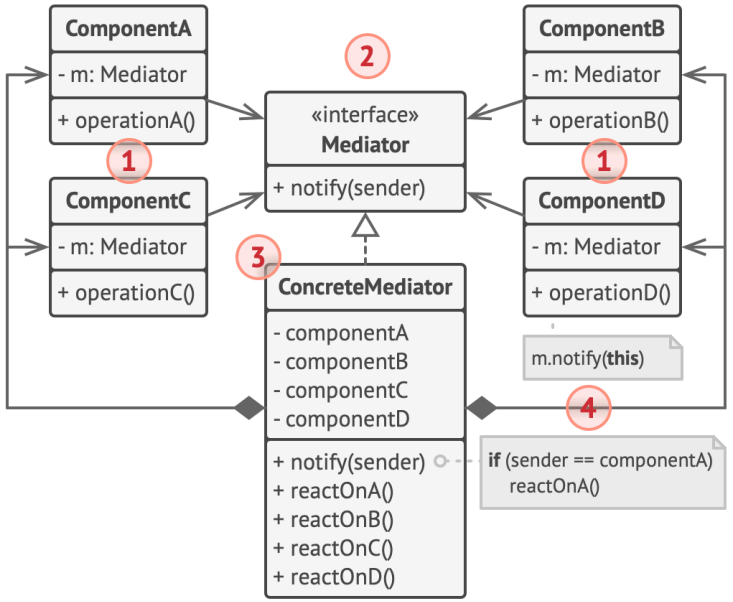




Behavioral Patterns
Mediator

- Bileşenlerin belirli olaylarının tetiklenmesi
- Bu olayların başka bileşenlerin durumunu değiştirmesi
- Direkt bağlantılarda bileşenlerin yeniden kullanılabilirliğinin ortadan kalkması
- Her bileşenin tek nokta üzerinden haberleşmesi

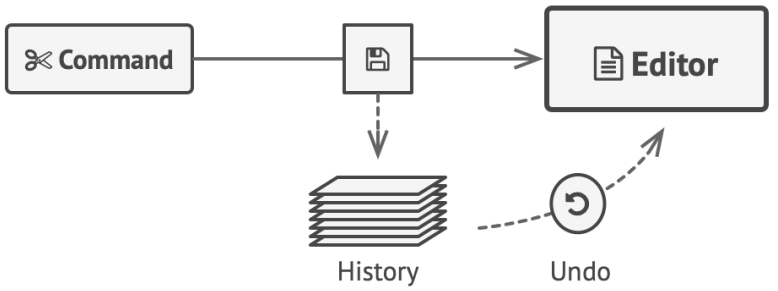




Behavioral Patterns

Memento

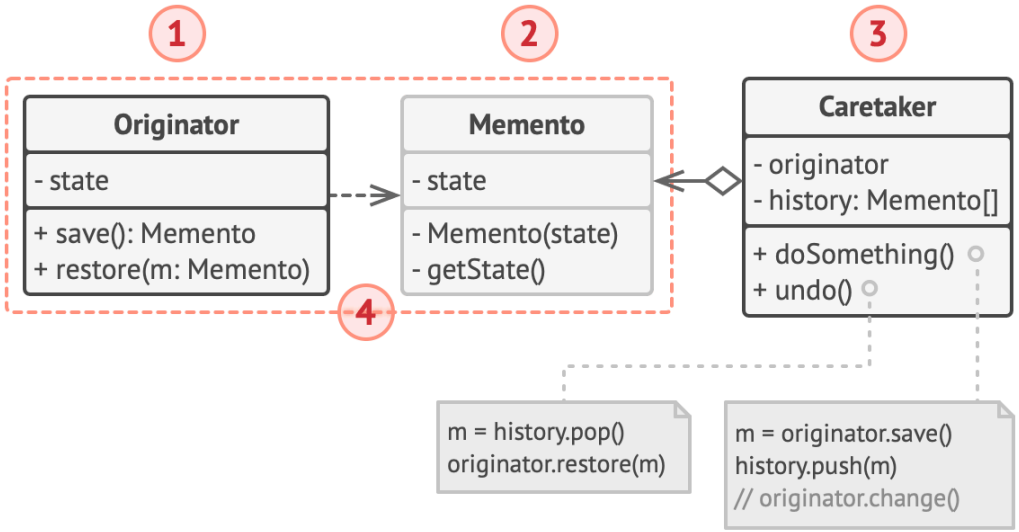
- Geçmiş durumların saklanması
- Hangi durumların saklanacağının belirlenmesi
- Bu durumların her nesne için ayrı ayrı toplanması
- Nesnelerin private alanları

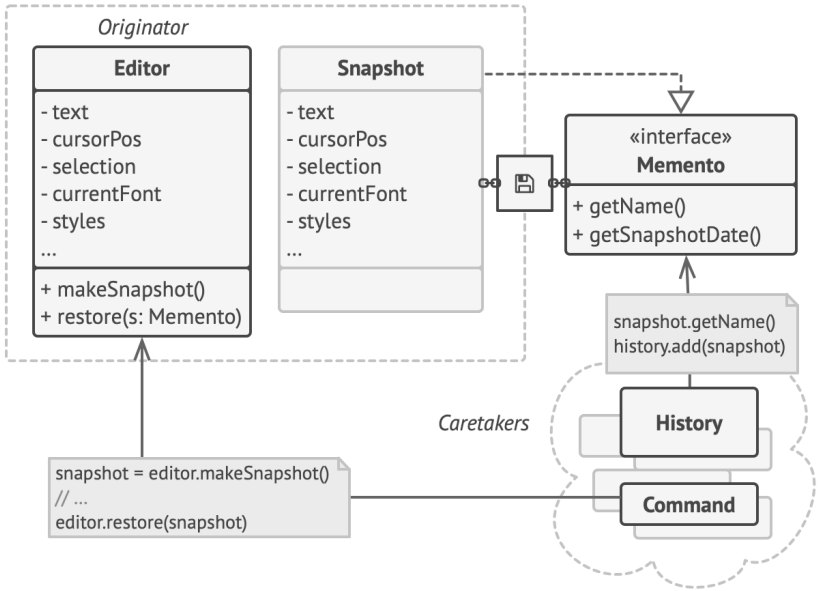


Behavioral Patterns

Memento

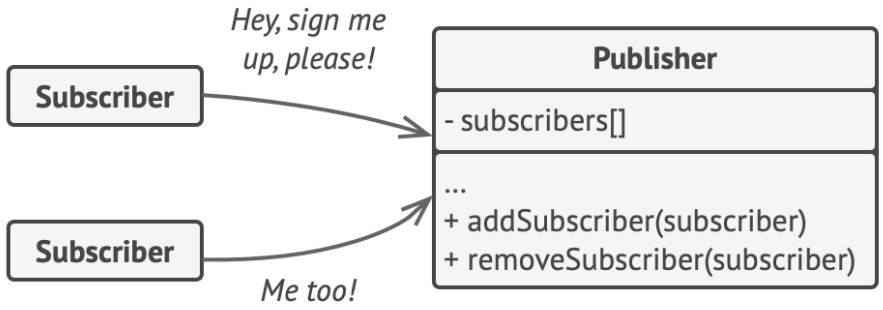
- Kopyalanması istenen sınıfların memento objesini içermesi
- Memento'nun sınıfların alanlarına erişebilmesi
- Memento'nun dışarıdan erişiminin sınırlanması





Behavioral Patterns
Observer

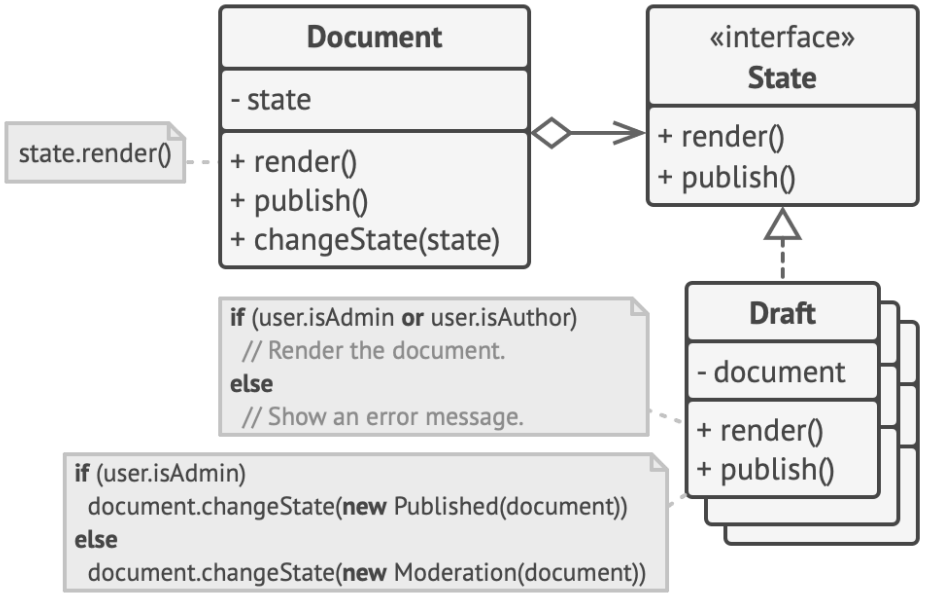
- Bildirim almanın maliyetli olması
- Her grubun her bildirimi istememesi
- Bildirim almak isteyenlerin belirli Publisher'lara eklenmesi
- Herkesin ihtiyaç duyduğu bildirimi alması



Behavioral Patterns

State

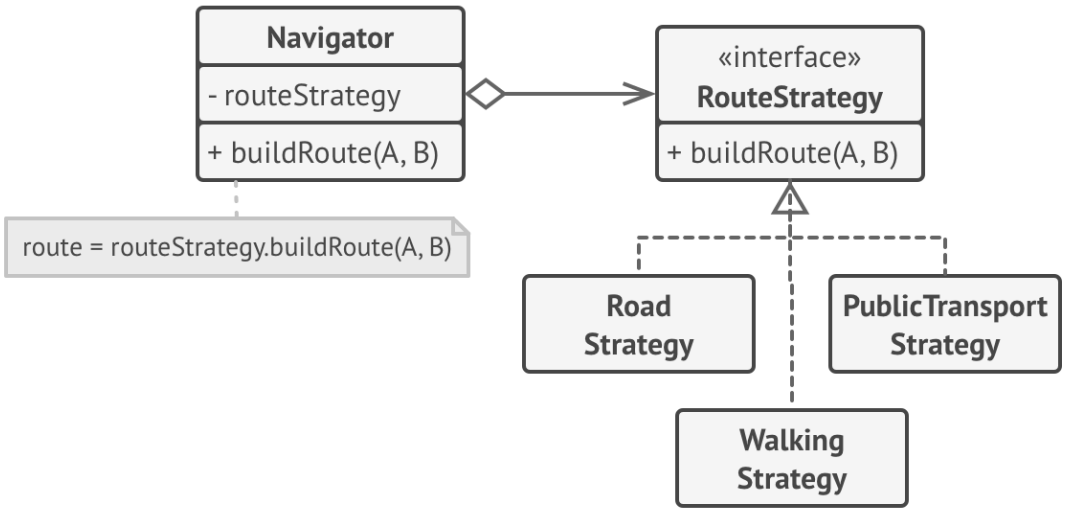
- Sonlu durum otomatları
- Durumlar
- Her durum için bir sınıf
- İçerik bir durum referansı tutar
- İçerik durumun değişmesi ile yeni nesneye bağlanır



Behavioral Patterns

Strategy

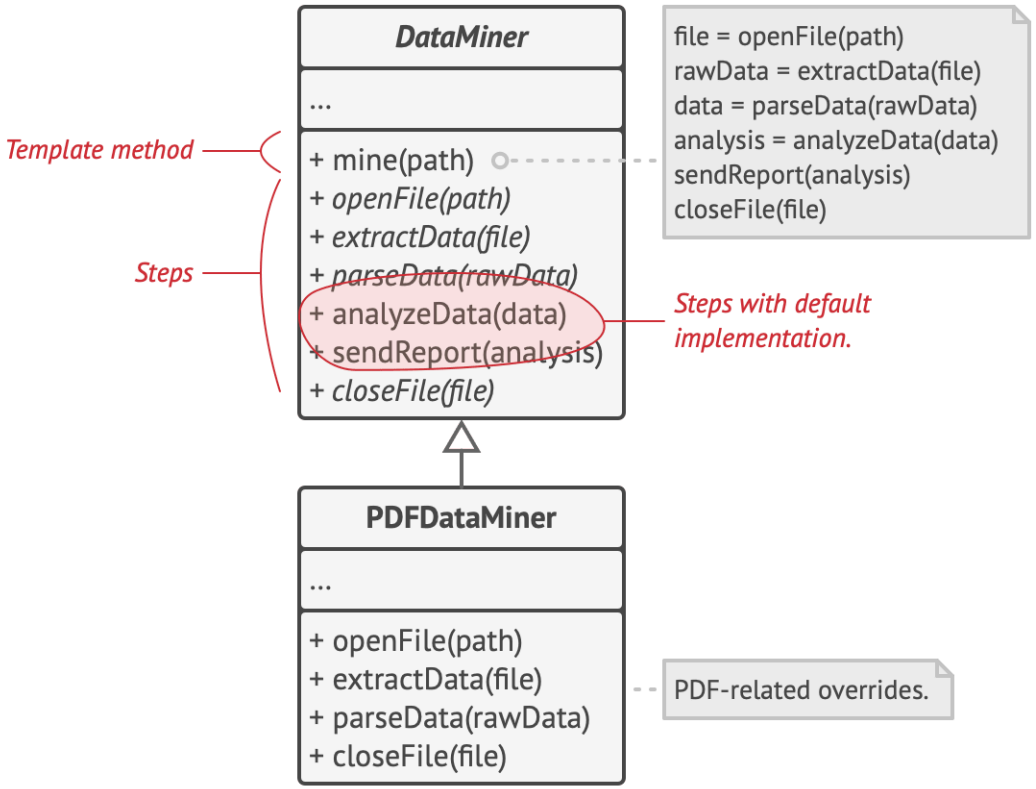
- Bir sınıfın yöntemlerinin belirli planlara göre farklı işler yapması
- Strateji kavramı
- Farklı stratejilerin aynı metotla kullanılması



Behavioral Patterns

Template Method

- Bir uygulamanın birden fazla adımı
- Aynı adımlar
- Farklılaşan adımlar
- Adımların bölünmesi
- Yalnızca ihtiyaç duyulan adımların alt sınıflar tarafından override edilmesi



Behavioral Patterns
Visitor

- Sınıfların yeteneklerinin geliştirilmesi
- Sınıfın modifiye edilmemesi
- Geliştirme işleminin sürekli olması halinde sınıfın bakım yapılamaz kadar genişlemesi
- Sınıfı parametre alıp bu sınıfla ilgili işi yapacak bir sınıf
- Visitor