**Command Pattern**

• **Intent**: Encapsulate a request as an object, thereby letting you parameterize clients with different requests, queue or log requests, and support undoable operations.

• **Amaç**: Bir isteği bir nesne olarak sarmala, böylelikle farklı isteklere, kuyruk ve log isteklerine sahip istemcileri parametrik hale getir ve geri alma işlerini destekle.

• Bu kalıpla iki şey çözülür:

• İstemciler, mesajı sadece hangi nesnenin aldığını bilmemekle kalmazlar aynı zamanda hangi metodun da mesajı yerine getirdiğinden habersiz olurlar.

• Yan metotlarla karmaşıklaşan nesne, kalıbın uygulanmasıyla daha odaklı hale gelir.

**PROBLEMS**

• Nesneler mesaj göndererek birbirlerinden istekte bulunurlar.

• Bazen istekte bulunan nesnenin, isteğin kendisini yani çağrılan metodu ve isteği gerçekte kimin yerine getireceğini yani metoda sahip olan nesneyi bilmemesi gerekir.

• Çünkü istemci ile muhatap olunan nesne ve işi yerine getiren metodu arasında olabildiğince az bir bağımlılık bulunması istenir.

• **Dolayısıyla istemcinin hem isteği kimin yerine getirdiğini hem de isteğin gerçekte hangi metotla gerçekleştirildiğini bilmemesi gerekir.**

**• Bu şunun gibi bir durumdur:**

• Bilmediğiniz birine gerçekte ne olduğunu bilmediğiniz bir işi, çok genel bir emirle, “yap” diye söylüyorsunuz ve o ne yapacağını bildiği için yapması gerekeni yapıyor.

• Buradaki “yap” emri çok genel bir emirdir, hernangi özel bir hareketi değil, sadece yapanın yapmasını bildiği şeyi temsil eder.

**PROBLEMİN DEVAMI ÖRNEK PROJE İCİNDE BULUNUYOR.**



**SONUCLAR**

• İsteyen ile isteği yerine getiren arasındaki bağımlılığı azaltmak ve isteği nesne olarak ifade etmek istediğinizde **Command** kalıbınım kullanın.

• **Command** nesneleri, isteği metot olarak soyutlamak yerine daha yetkin bir şekilde sınıf olarak soyutlayan nesnelerdir.

• Bu şekilde istek, tekrar yapılması ve geri alınması vb. ek davranışlarla beraber sınıf seviyesinde temsil edilebilir hale gelmiştir.

• **Command**, artan karmaşıklıktan dolayı, metot seviyesindeki soyutlamayı, sınıf seviyesine çıkaran bir kalıptır.

• Ne zaman sahip olunan durumda bir zorlukla karşılaşırsanız, ya yeni soyutlamalar oluşturun ya da var olan soyutlamaların üstüne çıkın.

• Bu size daha geniş hareket etme imkanı sağlayacaktır.

• Somut Command nesnesi isteği alınca farklı şekillerde davranabilir:

• Davranışı tamamen kendisi yerine getirebileceği gibi başka nesnelerden de isteyebilir.

• Redo/undo yanında emir listesi de desteklenebilir.

**DİĞER KALIPLARLA İLİSKİ**

• **Strategy**, bir algoritmayı soyutlarken **Command** bir isteği soyutlamaktadır.

• **Strategy’de** algoritmanın ismi bellidir ama gerçekleştirmesi farklılaşmaktadır.

• **Command’de** algoritmanın ismi de belli değildir bu yüzden execute gibi genel bir isim kullanmaktadır.

• Ama ikisi de karmaşıklığı, farklı nedenlerden dolayı da olsa, bir üst seviyedeki soyutlama ile aşmayı istemektedir.

• **Command**, **Composite** olabilir.

• Bu durumda bir istek, parçaları olan alt isteklerden oluşacak şekilde

ifade edilir.

• **Command**, **Memento** ile durumu saklayıp undo sonrasında geri

alabilir.

• **Command** nesnesi **Prototype** ile kopyalanabilir.

• **Command**, prosedürel dillerdeki geri çağırma (call-back) fonksiyonlarının nesne alternatifleridir.

• Dolayısıyla **Command**, **Observer** kalıbıyla birlikte kullanılabilir.

**UYGULAMA**

• Televizyon ile onu yöneten uzaktan kumanda arasındaki ilişkiyi

Command kalıbıyla modelleyip kodlayın.

• Kumanda üzerinde şu tuşlar vardır:

• Aç ve Kapat,

• Kanal seç,

• Sesi aç ve kıs.