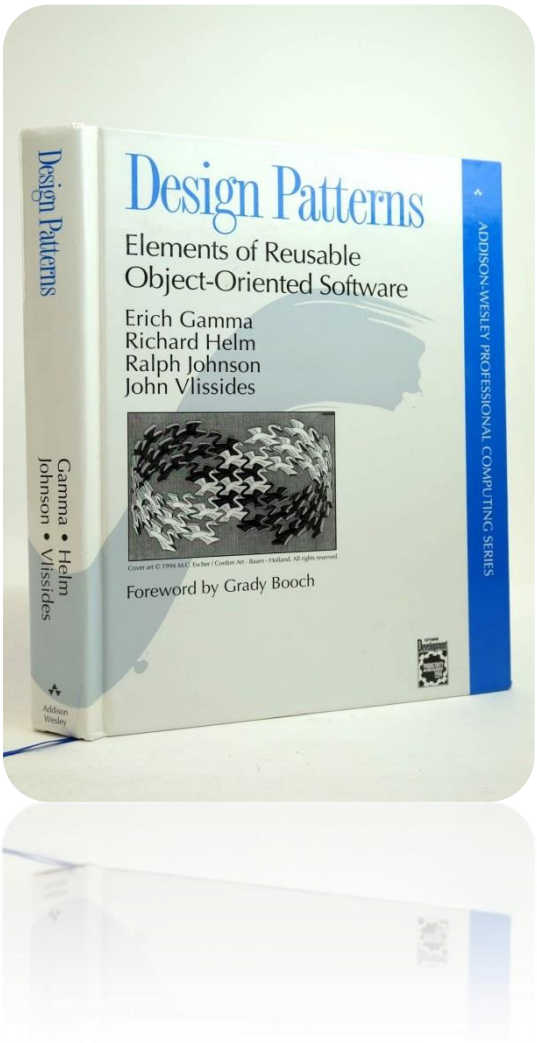
Gang of Four (GoF)



Yazılımlar daima değişir. Değişen şartlara ve artan ihtiyaçlara adapte olamayan yazılımlar ayakta kalamazlar. Üstelik bir yazılımın bakım malliyeti genellikle yazılımın geliştirilme ve yayınlanma sürecindeki diğer tüm maaliyetlerden daha fazladır. Bu durumda yazılımları meydana getiren bileşenlerin değişim taleplerine direnç göstermeyecek şekilde esnek, modüler, okunaklı olarak tasarlanması zorunludur.

Kısaca GoF (Gang of Four) tasarım kalıpları, sık tekrar eden bazı problemlerin çözümü için sunan yirmi üç adet şablon sunar. Bu tasarım kalıpları ilk kez ***Desing Patterns: The Elements of Reusable Object-Oriented Software*** isimli bir kitapta duyurulmuştur. Gang of Four ifadesi kitabın yazarı olan dört kişiye vurgu yapar.

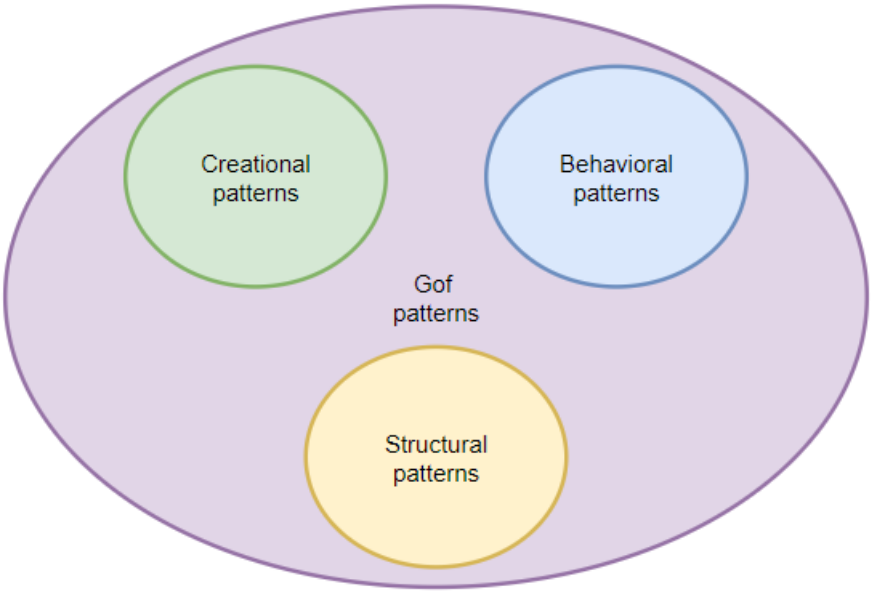
Tasarım kalıpları sık karşılaşın problemler için uygun tasarım yapılarını bizlere vermektedir. Bu tasarım kalıpları tecrübelerden edinimler sonucunda optimum tasarım yöntemlerini verir. Herhangi bir yazılımın tecrübelerden yola çıkarak elde edilmiş optimum tasarım şablonlarından meydana gelmesi uzun süreli bir kaç yarar sağlar.

1. Tasarım kalıpları belirli prensipler çerçevesinde oluşmuştur. Bu yüzden uygulamanın modülerlik düzeyini artırmak, yeniden düzenlenebilirliğini kolaylaştırmak gibi öenmli kazanımlar prensipleri ve şablonları uygulayarak mümkün hale gelir.
2. Yazılım dünyasında ortak bir dil olmuş tasarım yöntemlerini kullanmak bir çok geliştirici ile ortak bir dil kullanmak anlamına gelir. Bu sayede geliştirme ekibine sonradan katılanların konu hakkında aydınlanması kolaylaşır.

Kazanımla çoğaltılabilir. Ancak en önemli kazanım 1. maddede ifade edilenlerdir. Prensipler veya şablonlar yazılımların bakımı veya değişimi için gereken zaman, para ve enerjiyi azaltmak için ortaya atılmıştır.

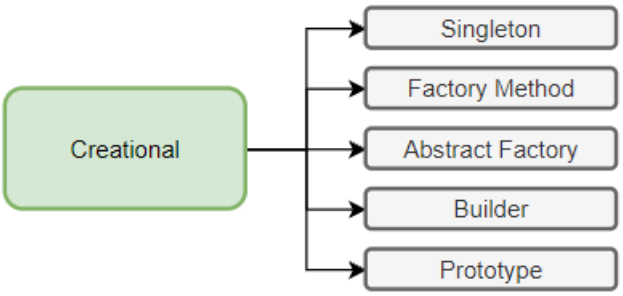
GoF Kataloğu

GoF sistematiğindeki tasarım kalıpları bu kalıpları hedeflerine yönelik olarak üç farklı küme altında yer alır. Bu kümeler nesnelerin oluşturulması ile ilgili kalıpları kapsayan ***Creational***, belirli bir problemin çözümüne yönelik bir davranış yerine getiren kalıpları kapsayan ***Behavioral*** ve yine belirli bir problemi ortadan kaldırmak için bir yapı kurmayı amaçlayan kalıpları kapsayan ***Structural*** kümeleridir.



Creational grubu

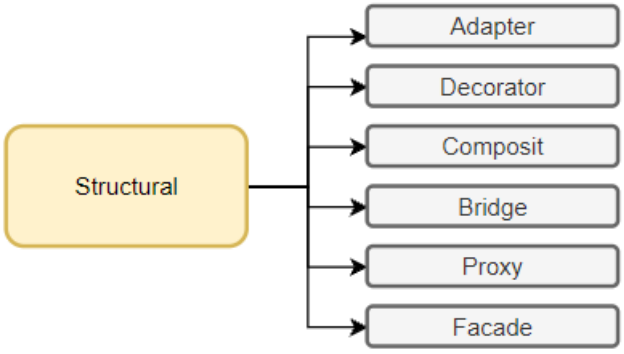
Object-Oriented dünyasında yazılımlar çelitli nesnelerin birbirlerinin fonksiyonlarını çağırmak suretiyle çalışırlar. Yazılım geliştiren ekipler nesnelerin tasarımını kaynak kodlar içerisine koydukları sınıflarla yaparlar. Bu sınıflardan nesne oluşturulması kimi zaman düşünüldüğü kadar basit olmayabilir. Creational grubunda bulunan kalıplar sınıflardan nesne oluşturmak üzere tasarım örüntüleridir. Bu grupta beş adet tasarım kalıbı yer alır.



Örneğin ***`Factory Method`***, bir ürün ailesinden(bir interface veya abstract sınıftan meydana gelmiş) nesne oluşturmayı amaçlar. Eğer ürün ailesinin üyesi olan sınıfların oluşturulmasına dair detaylar gizlenmek isteniyorsa veya bu detaylar kompleks bir yapıda ise nesne üretmek için istemci tarafa makul bir yapı sunmayı amaçlar. Bu sayede kaynak kod karmaşık nesne oluşturma operasyonlarından arındırılmış olunur.

Structural grubu:

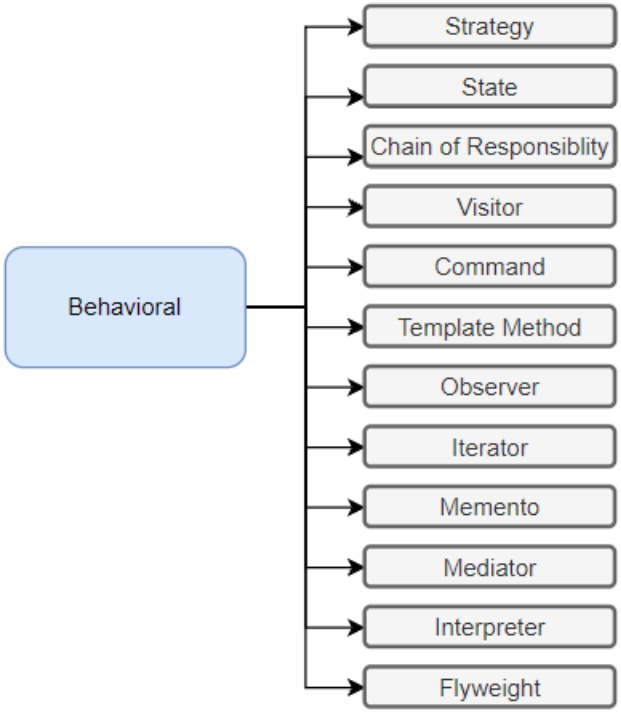
Bir problemin Object-Oriented dünyadaki optimum çözümü bazen o problemi uygun bir yapıyla ifade etmekle mümkün olur. Structural grubundaki tasarım kalıpları bilinen bazı problemlerin en iyi çözümünü ortaya koyabilmemiz için bazı yapılar tasarlarmayı önerir. Bu grupta altı adet tasarım örüntüsü vardır.



Bu grup için ***Adapter*** iyi bir örnek olabilir. Yazılım sistemleri bazen sonradan kullanılan bileşenler yüzünden mevcut sistemle uyuşmayan komponentler ile karşı karşıya kalır. Bu durumda mevcut sistemle uygun hale getirilmesi gereken bir yabancı varlık ile karşı karşıya kalınmıştır. Adapter, optimum şekilde yabancı bir bileşenin mevcut sisteme uygun hale getirilmesi için ilgili yabancı öğeyi sarmalamaya dair bir yöntem sunar. En nihayetinde ortaya çıkan yeni bileşen(Adapter bileşeni) yabancı bileşeni mevcut sisteminize uyumlu hale getiren bir yapı oluşturmuştur.

Behavioral grubu:

İyi bir tasarım için kimi zaman bazı davranışlara ihtiyaç duyulur. Bazı problemler o problemi en iyi çözümü için bir davranışın sisteme kazandırılmasına ihtiyaç duyar. Davranışsal kalıplar sistemin belirli bir davranışı yerine getirmesi için uygun tasarım örüntüleridir. Optimum tasarımlar yapabilmek için kimi problemleri çözerken uygulayabileceğiniz on iki davranışsal tasarım kalıbı vardır.



Anlaşılması en basit davranışsal kalıplardan birisi Iterator olarak karşımıza çıkar. ***Iterator***, bir veri yapısı içinde yer alan elemanların dış kaynaklara iletimini uygun bir yapıda gerçekleştirir. Her yazılımcı veri yapılarını kullanır. En nihateyinde bir döngü vasıtasıyla veri yapısı içindeki elemanlara ulaşmak genellikle bir Iterator pattern uygulaması ile yapılmaktadır.