**BUILDER PATTERN**

**Intent**: Separate the construction of a complex object from its representation so that the same construction process can create different representations.

• **Amaç**: Karmaşık bir nesnenin oluşturulmasını sunumundan ayır öyle ki aynı oluşturma süreci farklı sunumlar yaratabilsin.

**PROBLEM**

Sıklıkla nesneler dolayısıyla yaratılmaları karmaşıktır.

• Böyle karmaşık bir nesneyi yaratmak, basit bir kurucu (constructor) çağrısından ziyade bir süreç içinde olur.

* Çünkü karmaşık nesnelerin çok ve karmaşık parçaları vardır.
* Yaratma süreci farklı yollar izleyebilir, farklı safhaları olabilir ve bu yol ve safhalar farklı isimlere sahip olabilir,

• Nesne için zorunlu ve seçimli parçalar söz konusu olabilir vs.

* Ve istemci bunlarda tercih belirtmek isteyebilir.



• Bir nesnenin karmaşık olduğunun en temel göstergesi çok parametre alan constructorlardır.

• Telescoping constructor anti-pattern (uzun kurucu anti-patternı)

• Varsayılan kurucu sağlıyorsa bu durumda pek çok set metodu da vardır.

• Kurucuya ya da set metotlarına geçilen parametrelerin de oluşturulması

gerektiği düşünüldüğünde, bu sürecin yönetilmesi gerektiği açık hale gelir.

• Çünkü nesnenin yaratılması zor ve hataya açıktır.

**Bu türden nesneleri oluşturmak için gerekli süreç Builder kalıbı ile modellenebilir.**

**• Bu kalıpta bir anda yaratmak (create) yerine inşa söz konusudur.**

**• Bu sebeple builder kelimesi ile süreçsellik vurgusu yapılmıştır.**

GoF, bir RTF (Rich Text Format) dokümanından ASCII vb. farklı formatta

text dokümanlarına çevirecek (conversion) bir TextConverter yapısından bahseder.

• Dokümanın boşluk, kelime, vb.farklı parçaları (token) vardır.

• Aynı kaynak dokümandan çevirim ile farklı sunumlar (representation) yani

farklı formatlarda yeni dokümanlar elde edilmektedir.

• Yani farklı parçaları kullanan ve farklı çevirim yapan değişik algoritmalar

söz konusudur.

* İstemci, farklı çevirim şekillerinden ve parçaların oluşturulmasından

bağımsız olmalı, çevirimi gerçekleştiren algoritmayı bilmemelidir.

• İstemciyi, bir nesnenin yaratılmasından bağımsız kılmak amacıyla Abstract Factory/Factory Method ya da Booch’un statik factory metotlarının nasıl kullanılacağını biliyoruz.

• Fakat buradaki problem, sadece yaratmayı gilemek değil, yaratmayı sürece çevirmek ve gerekirse istemciyi süreçteki tercihleriyle ilgili olarak yönlendirmek.

**Ama sadece Factory Method ve Abstract Factory bu problemi çözmez.**

• **Çünkü J. Bloch’un da Effective Java kitabında söylediği gibi bu iki kalıp da kurucular gibi büyüme (scale) problemi olan yapılardır.**

• Dolayısıyla nesnenin bina edilme süreci bir arayüzün farklı gerçekleştirmeleri olarak kurulur.

**• Bu kalıpta vurgulanan şey, nesnenin nerede yaratılacağı değil, nesnenin yaratılma sürecidir.**

**SONUCLAR**

Builder kalıbı ile oluşturulan nesne yaratma süreci bir arayüzde ifade edildiğinden tekrar kullanılabilir.

• Dolayısıyla aynı süreç farklı nesne görünümlerini destekler.

* Örneğin ComputerBuilderI’ın farklı gerçekleştirmeleri farklı CPU, diplay vb. parçaları kullanarak farklı kalitede ve süreçte Computer nesneleri bina eder

• Bu süreç, karar mekanizmaları vb. algoritmik detayları da içinde barındırır.

**KULLANIMLAR**

**java.lang.Appendable arayüzünün gerçekleştirmeleri:**

**• java.lang.StringBuilder ve java.lang.StringBuffer**

**• java.nio.Buffer alt sınıfları**

**• javax.ws.rs.core ve javax.ws.rs.client paketlerinde**

**DİĞER KALIPLARLA İLİŞKİSİ.**

**Builder** kalıbının amacı, karmaşık olan nesne yaratma sürecini yönetmektir.

• **Factory Method ve Abstract Factory** kalıpları ise sadece nesne yaratma sorumluluğunu devir almak içindir.

• Bu iki kalıp ile **Builder** birbirlerinin alternatifi değildirler.

**• Factory Method ve Abstract Factory kalıpları Builder kalıbını kullanabilir.**

**UYGULAMA**

**Bir bisikletin şu parçaları mevcuttur:**

• Frame,

• A pair of wheels, front and rear

• Front set

• A pair of pedals, left and right

• A pair of brakes, front and rear

• Chain



Buna göre bir bisiklet nesnesinin oluşturulması sürecini **Builder** kalıbı ile modelleyip kodlayın