GRASP Kalıpları

GRASP (General Responsibility Assignment Software Patterns) nesnelere sorumluluk atamada yol gösteren temel kalıpların genel adıdır.

1. Bilginin Uzmanı

Sınıflara sorumluluk atamanın temel prensibi nedir?

Bir sorumluluğu bilginin uzmanına, yani onu yerine getirecek veriye sahip olan sınıfa atayın.

Sale → Sales Lıne Item (bu bilgiler sale'da var)

2. Yaratıcı

Bir sınıftan nesne yaratma sorumluluğu kime aittir?

B, A nesnelerini içeriyorsa
B, A nesnelerinin kaydını tutuyorsa
B, A nesnelerini yoğun olarak kullanıyorsa
A'nın yaratılması aşamasındaki başlangıç verilerine B sahipse
Bu koşulları sağlayan birden fazla nesne varsa, sorumluluk nesneyi içeren sınıfa verilmelidir

Register → Sale (sale register'ın üyesi)

3. Az Bağımlılık

Diğer sınıfların değişimlerinden etkilenmeme, tekrar kullanılabilirlik nasıl sağlanır?

Sorumlulukları, sınıflar arası bağımlılıklar daha az olacak şekilde sorumlulukları atayın.

Register → Sale → Payment (bağımlılık daha az)

4. Denetçi

Sistem olaylarını arayüz katmanından alıp ilgili nesnelere yönlendirmekle kim sorumludur?

Görüntü veya senaryo (oturum) denetçileri kullanılır.

SaleJFrame -- Register – Sale (register denetçisi)

5. İyi Uyum

Karmaşıklık nasıl idare edilebilir?

Sorumlulukları sınıf içinde iyi bir uyum olacak şekilde atayın.

Register → Sale → Payment (kim oluştursun)

6. Çok Şekillilik

Tiplere bağlı alternatifler nasıl ele alınmalı? Sisteme kolay monte edilebilen yazılım parçaları nasıl gerçeklenir?

Birbirleriyle ilgili alternatif davranışlar, tiplere bağlı olarak değişiklik gösteriyorsa bu davranışları çok şekilli metodlar kullanarak gerçekleyiniz. Koşullu deyimler kullanmak iyi bir yöntem değildir.

ItaxCalculatorAdaptor (farklı adaptörler aynı başlıkta toplanmış)
GenericShape (Line, Rectangle, Circle, ...)

7. Yapay Sınıf

Uzman kalıbının önerdiği çözüm, iyi uyum ve az bağımlılık kavramları ile çelişiyorsa sorumlulukları hangi sınıfa atamak gerekir?

Eğer sınıflararası bağımlılıkları azaltıyorsa ve yazılımın tekrar kullanabilirliğini arttırıyorsa, birbirleriyle uyumlu sorumlulukları gerçek dünyada var olmayan yapay bir sınıfa atayabilirisiniz.

PersistantStorage, Connection, ...

8. Dolaylılık (Arabirim)

Özellikle kolay değişebilen iki birim arasındaki bağımlılığı azaltmak için sorumluluklar nasıl atanmalıdır?

İki birim arasındaki bağımlılıkları azaltmak için sorumlulukları bir ara nesneye atayın.

Sale → TaxMasterAdapter → TCP Socket Connection

9. Değişimlerden Korunma

Nesneler, sistemler ve alt sistemler, bu biirmlerdeki değişim ve kararsızlıklar, birbirlerini en az etkileyecek şekilde nasıl tasarlanmalıdır?

Sistemde değişimlere açık, kararsız noktaları belirleyin ve bu değişim noktalarının etrafında kararlı bir arayüz oluşturacak şekilde sorumlulukları atayın.

Vergi hesaplamada değişik programlar kullanma

Yabancılarla Konuşma

Bir nesnenin mesaj gönderebileceği nesnelere sınırlamalar getirmektedir. Kendisi, metodunun parametresi olan nesne, nesnenin niteliği olan nesne, metodun içinde yaratılan nesne durumları dışında erişilmemelidir.

Money amount = sale.getPayment().getTenderedAmount(); // KULLANMAMAYA ÇALIŞ

GoF Tasarım Kalıpları

GoF (Ganf of Four) kalıplarıdır. Piyasada en çok kullanılan ve probleme özgü kalıplardır.

1. Adaptör

İstenen işleri yapan fakat farklı arayüzleri olan benzer birimler için tek bir kararlı arayüz nasıl yaratılır?

Birimin orjinal arayüzünü bir adaptör nesnesi kullanarak başka bir arayüze dönüştürün.

Circle -- XXCircle
Sale -- TaxMasterAdapter -- TaxMasterAdapter

2. Fabrika

Nesnelerin yaratılmasında karmaşık kararlar vermek gerekiyorsa ve uyumluluğu arttırmak için yaratma işlemlerinin diğer işlemlerden ayrılması isteniyorsa nesne yaratma sorumluluğu nasıl atanmalıdır?

Nesnelerin yaratılma sorumluluklarını yapay bir sınıf olan fabrika sınıfına atayın.

ServicesFactory
SoyutFabrika: Birbirleriyle ilişkili birden fazla nesne durumu. Fabrika adaptörü diyebiliriz.

3. Tekil Nesne

Bir sınıftan sadece tek bir nesne yaratılması isteniyor. Bu nesne diğer nesnelere global tek bir erişim noktası sağlayacak.

Sınıfın içine tekil nesneyi yaratıp adresini döndüren statik bir metod yerleştirin.

getInstance()

4. Strateji

Birbirleriyle ilgili olan fakat farklılık gösteren algoritmalar nasıl tasarlanmalıdır? Bu algoritmalardaki değişimin sistemi etkilemesi nasıl önlenmelidir?

Her algoritmayı ayrı bir sınıfın içinde gerçekleyin. Bu sınıfların arayüzleri aynı olmalıdır.

SalePricingStrategy (PercentDiscountPS, AbsoluteDiscountPS, ...) [adaptöre çok benziyor]

5. Bileşik Nesne

Nesne grupları veya bileşik nesneler, tek bir nesne gibi çok şekilli olarak nasıl tasarlanır?

Bileşik ve atomik nesneleri aynı somut sınıtan türetin. Bileşik nesnelerin içine atomik nesneleri içeren bir liste yerleştirin.

CompositePricingStrategy(pricingStrategies)

6. Ön Yüz

Değişik arayüzlere ve yapılara sahip hizmetleri içeren karmaşık bir alt sistem olsun. Alt sistem ileride değişebilir ve alt sistemin sadece bazı olanaklarını kullanmak istiyorum. Bu alt sistem ile bağlantı nasıl sağlanmalı?

Alt sistem ile bağlantıyı sağlayan bir ön yüz nesnesi tanımlayın. Ön yüz nesnesi alt sistemdeki tüm hzimetler için tek bir ortak erişim noktası oluşturacaktır.

Adaptöre çok benziyor, en temel farkı karmaşık sistemlerdeki bazı fonksiyonlara ulaşılıyor. package, namespace kavramlarını da buna benzetebiliriz.

7. Gözlemci

Değişik tipte abone nesneleri, yayıncı nesnelerdeki durum değişiklerini sezmek ve bu değişimlere kendilerine göre tepki göstermek isterler. Yayıncı nesnenin, abone nesnelere daha fazla bağımlı olması istenmez.

Abone nesnelerini ortak bir üst sınıftan türetin. Yayıncı nesne isteyen aboneleri kendi listesine kayıt eder. Yayıncı kendi durumunda bir değişiklik olduğunda listesindeki abonelere bu durumu bildirir.

PropertyListener (propertyListeners) → onPropertyEvent
AlarmClock (alarmListeners) → onAlarmEvent

8. Köprü

Nesneye dayalı tasarım yapılırken sistemin ileriki aşamalarda karmaşıklaşması nasıl önlenebilir?

Soyutlama ile gerçeklemeyi birbirinden ayrı tutun, böylece ikisini birbirinden bağımsız olarak değiştirebilirsiniz.

Shape (Rectangle, Circle, ...) <> ----------> Drawing (V1Drawing, V2Drawing, ...)

9. Dekoratör

Bir nesnenin davranışı bir dizi işlemden oluşuyorsa ve bu işlemlerin sırası ve sayısı belli koşullara göre değişiyorsa strateji kalıbı gerekli çözümü sağlayamaz. Ne yapılmalı?

Soyut dekaratör sınıfından türeyen nesneler, program çalıştılırken ortak bir listeye yerleştirilir.

Component (SalesTicket, TicketDecorator(next, HeaderDecorator, FooterDecorator))