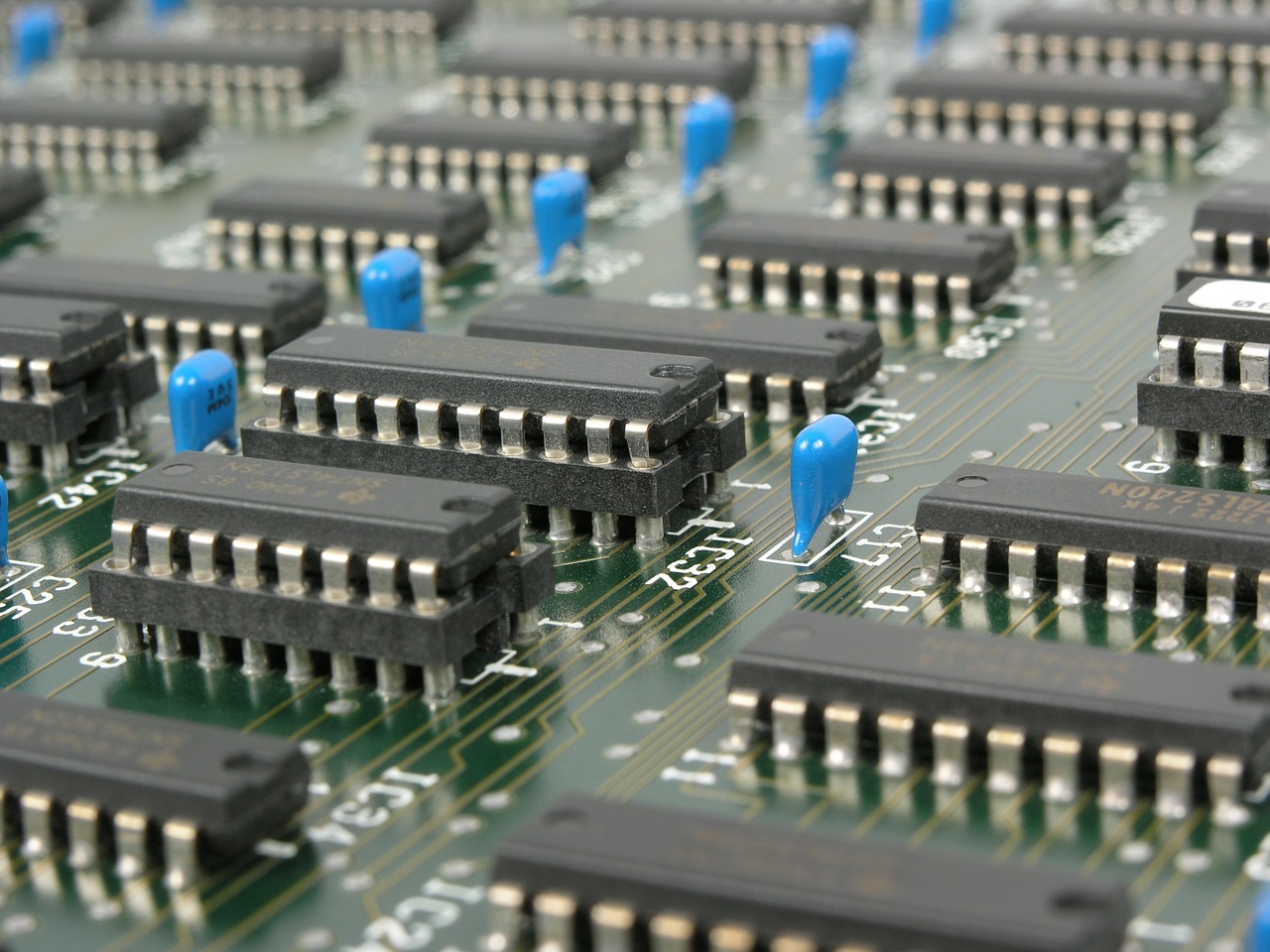
Loose Coupling – Gevşek Bağlılık Prensibi



Merhaba sevgili tasarım prensiplerine meraklı dostlarım.

Bu yazımda SOLID tasarım prensiplerinin içinde olmayan fakat Enterprise tasarım prensipleri içerisinde sayabileceğimiz Loose Coupling (Gevşek Bağlantı) İlkesinden bahsedeceğim. Böylece Interface’e daha farklı bir gözle bakabilir ve ISP (Interface Segregation Principle) ile DIP ([Dependency Inversion Principle](https://www.turkayurkmez.com/dependency-inversion-principle/)) prensiplerindeki kavramlara daha net bir yaklaşım sergileyebiliriz.

Loose Coupling ilkesini anlamak için “sıkı bağlantı” nın ne olduğunu anlamak gerekiyor sanırım. Bir sistem; içerisinde bulunan mekanizmanın çalışabilmesi için belirli bir nesneye mutlaka ihtiyaç duyuyorsa, o nesneye sıkı sıkıya bağlıdır. Bir gerçek hayat örneği ile bu kavramı anlatmaya çalışalım. Eğer iş dünyası personeline “Sıkı Bağlı” olsaydı o zaman iş ilanında direk personelin adı olması gerekirdi (düşünsenize bir iş ilanında “Adınız Soyadınız aranıyor yazdığını  “)! Oysa günümüzde iş ilanlarında sadece aranan yetkinlikler (yani Interface’ler) bulunuyor. Haliyle, bu yetkinlikleri taşıyan (bu interface’i implemente eden) kişiler ilgili ilana başvurup değerlendiriliyor.

Yani, bir organizasyonda (çoğu Hollywood filminde olduğu gibi) “bu işi yalnızca bay/bayan X yapabilir” cümlesi kuruluyorsa, o organizasyon X kişiyle “sıkı bağlantı” halindedir. Oysa bu cümle, şu biçimde kurulsaydı çok daha sağlıklı olurdu; “bu işi, şöyle yetkinliğe sahip herkes yapabilir”. İşte Loose Coupling ilkesinin öncelikli görevi de sistemi bu hale getirmektir.

Tam da bu kısımda, bu kavramın yaratıcısı Karl E. Weick’den bahsetmemek olmaz. Kendisi bir “Kurumsal Psikoloji uzmanı” ve hayatı bu alanda yaptığı çalışmalarla geçmiş bir teorisyen. 1975 yılının Şubat ayında Ulusal Eğitim Enstitüsü (National Institue Of Education – NIE) konferansında, “Eğitim Kurumlarında Gevşek Bağlantı Sistemi” isimli bir sunum yapar (meraklısı [buradan](http://www.education.umd.edu/EDPA/courses/EDPL744-06/2.27.06%20Weick%20Loosely%20Coupled%20Systems%201976.pdf) ulaşabilir). İşte Loose Coupling ilk kez burada bu sunumda dile getirilir.

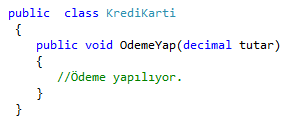
Bu ayrıntıyı burada belirtmemin sebebi şu; bugün büyük ölçekli yazılım projelerinde, işlerimizi büyük ölçüde standartlara kavuşturan (SOLID gibi) prensiplerin temelleri aslında bunun gibi sosyolojik, psikolojik veya diğer mühendislik alanlarında atılıyor. Yaşamakta olduğumuz dünyada, her şey birbirini o kadar makro düzeyde etkiliyor ki… Tek yapmamız gereken inovasyona her zaman açık olmak ve gözlemleyebilmek.

Peki, bu ilkeyi yazılım dünyası nasıl uyguluyor? Gelin bunu da bir örnekle görelim. Senaryomuz, bir ödeme işlemi üzerine olsun.

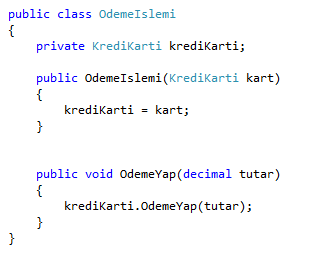
Tight Coupling (Sıkı Bağlantı) Örneği

Örneğimizde, KrediKartı isimli bir sınıfımız olsun. Bu sınıfın örneğini (instance) kullanan bir de OdemeIslemi sınıfımız var. Kodları ise şu şekilde:

Kredi Kartı sınıfı:



Ödeme işlemlerini üstlenecek olan sınıf:

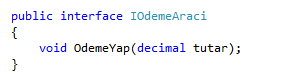


Gördüğünüz gibi OdemeIslemi sınıfım oluşturucu metodunda (constructor) bizden KrediKarti nesne örneği istiyor. Daha sonra bu nesneyi, OdemeYap() metodunda kullanıyor. Burada, projenin ilerleyen safhalarında başka ödeme seçeneklerinin de gelebileceğini düşünürseniz; mimarinin oldukça sıkıntılı olduğunu göreceksiniz. İşte, size bir “sıkı bağlantı” örneği.

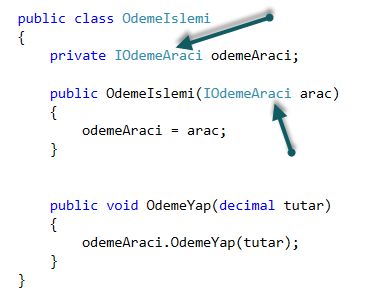
Hadi şimdi gelin bu bağlantıyı gevşetelim.

Loose Coupled (Gevşek Bağlantı) Örneği

Mademki gelecekte ödeme şekli değişiklik gösterecek, o halde buna göre bir önlem almamız gerekiyor. OdemeYap() metodunun, tüm ödeme araçlarında (PayPal, Kredi Kartı, SMS) bulunmasının şart olduğunu sanırım fark etmişsinizdir. Öyleyse kolları sıvayalım ve OdemeYap metodunu içeren bir Interface oluşturalım:

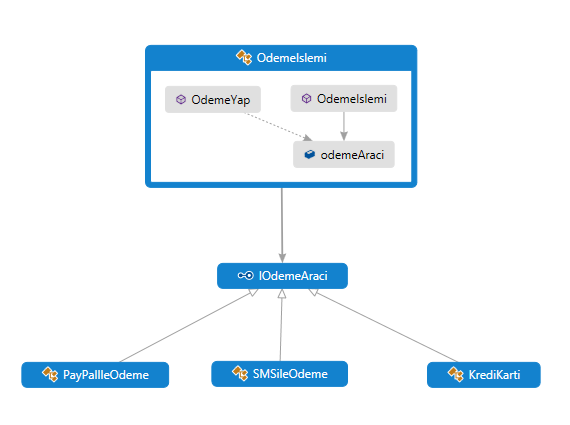


Pekâlâ, şimdi de OdemeIslemi sınıfımı bu mimariye göre tekrar yazıyorum:



Dikkat ettiğiniz gibi yaptığım tek değişiklik spesifik bir sınıf örneği yerine; parametre olarak IOdemeAraci interface’ini kullandım. Böylelikle; bu interface’i implemente eden tüm sınıfları bu metoda parametre olarak atayabileceğim. Doğal olarak; ilerleyen zamanlarda gelecek her yeni ödeme biçimi için OdemeIslemi sınıfında değişiklik yapmama gerek kalmayacak.

Son olarak; bu ilişkiyi grafik ile incelersek aşağıdaki çıktıyı göreceğiz:



Evet dostlar… Nesne Yönelimli Tasarım konulu bir makalemizin daha sonuna gelmiş bulunmaktayız. Sağlıcakla kalın.