**DEPENDENCY INJECTION**

**Coupling and Dependency**

**Coupling**, daha genel bir isim nesneler arasında n tane nesne arasında birbirleri ile alakalı bağımlılığın en genel ismi coupling, bu anlamda daha bilimsel bir isim.

Coupling **aslında bağlanmanın derecesi** ile alakalı konustugumuz zaman genel olarak couplingden bahsederiz.

**Dependency’I** de bağımlılık diye çeviriyoruz. Dependency ama daha özel bir isim. İki nesne arasındaki bağımlılık birisindeki değişim muhtemelen diğerinde de değişime zorlar dolayısıyla bu anlamda dependency dediğimiz şey direct coupling yani doğrudan iliski.

Birden fazla nesne aynı collection, library’I kullanıyor olabilir dolayısıyla bunları kullanmalarından dolayı nesneler arasında direct coupling vardır. Çünkü o ortak kullanılan yapı değişirse hepsi değişir. Hepsinin arasında bir coupling var ama dependency yok.

Dependency ortak olan yapıya doğru bağımlılığa **dependency** deriz. Ama o bütün nesneler arasında bir coupling vardır, third party yapının üzerinden.

**Example**-

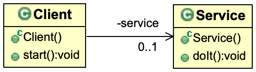


A ve B’nin C üzerinden bir couplingi var. Çünkü ortak yapıyı kullanıyor. Ama dependency olarak bakacak olursak A’nın C’ye, B’nin C’ye Dependency’si var. A ile B arasında bir dependency yok bu anlamda.

Yani A ile B arasında C üzerinden coupling.

**Birinde bir değişiklik varsa diğerinde de bir değişiklik olma ihtimali yüksekse o zaman bir dependency vardır diyebiliriz.**

**DEPENDENCY**

 **exampleone image.**

**Farklı Dependency tipleri var. Üç çeşit: Abstraction, Realization, Usage**

**Abstraction :**

The client and the supplier represent the same concept at different levels of abstraction or from different viewpoints.

**Client ile supplier aynı kavramı farklı soyutlama ve bakış açısı ile sunuyorlar. Bakış açısı da yukarı cıktıkca generic, Parentta childe dogru indikce daha spesifik halleri olur.**

**Realization:**

Realization is a specialized Abstraction dependency between two sets of objects, one representing a specification (the supplier) and the other representing an implementation of that specification (the client).

**Dependency’nin daha özel bir halidir. Objeler arasında supplier’ın olması gerekeni belirlemesi ama client’inde olması gerekeni de yerine getirmesi. Dolayısıyla birisi specificationı(supplier) yani olması gerekeni tarif ediyor, diğeri ise (client) bunu sağlıyor. Yukarıdaki supplier asagıdaki client oluyor, biraz daha asagıdan yukarı dogru implementi düsündügümüz zaman.**

**USAGE:**

The Client requires the service for its full implementation or operation.

**Genel olarak kullanma iliskisi. Herhangi bir sekilde bir objenin diğerini kullanma iliskisi.**

** örneği DependencyTypesExample paketi in**

**Abstraction and Realization**

Bu kavramlar daha pratik sekilde is-a iliskisi dedigimiz sey.

**Is-a** **inheritance**, **sub-typing ya da sub-classing** seklinde de söylenebilir.

**Abstraction’da**. client, parent’in daha spesifik halini sunar, Parent ise Client’in daha genel ve daha generic halidir.

**Realization’da** bir implementation parent’ta, supplier’da tamamen detayıyla tarif edilmemis,sadece neligi yani olması gerekeni söylemis bir yapıyı, davranısın nasıllıgını, calısacak sekilde yapmak **dolasıyla bir interface’I implemente etmek anlamına geliyor.**

**Yani Abstraction’da implementation inheritance javada(extends) sekilde yapılması,**

**Realization’da ise interface’I implemente etmek diyoruz.**

**USAGE DEPENDENCY**

Client service olmadan incomplete yani service olmadan client calısamaz.

Service’a ihtiyac duyma iliskisi iki sekilde olabilir, bazen definition ya da implementation seklinde.

Yani



Client olarak calısmak icin sadece bir isi yerine getirmek icin bir baska nesneye yani service ihtiyacımız varsa mesela g’nin calısması icin D’nin var olması gerekiyorsa bu bizim implementationa yani bir usage dependency’e ihtiyacımız oldugu anlamına geliyor.

Definition icin bağımlılık biraz daha güclü bir bağımlılık cünkü definitiondan kasıt A class’ıı bütün detayıyla tanımlayabilmek icin sadece davranısı değil, implementation detaylarını da vermek gerekiyor. Instance variable olarak B’ye sahip olmak al-kullan gibi değil daha derinden bir dependency’e sahip olma anlamına gelir.

**Yukarıdaki örnekte implementation iliskisi g() ile var, definition iliskisi B ile var.**

****

Eğer client bir baska class’ın nesnesine kendi parcası olarak ihtiyacı varsa o zaman dependency definition icin demektir. Yukarıdaki kodda (private B b)

Classı tam olarak tanımlamak icin demektir ve daha güclü bir iliski oldugu acıktır.

DependencyTypesExample paketi icinde örnek.

**Association’ın iki formu var.**

1.si Sadece bir araya getirme. Biraz daha gevşek durum. Yani **Aggregation**.

2.si ise bunun daha güclü hali **Composition**.

**Örnek verecek olursak**

**Bilgisayar Aggregation’a örnek cünkü pc olmadan cpu, ram, disk etrafta olabilir. Yasam döngüsüne devam edebilir.**

**Universite ile fakülteler arasında compositiona örnektir, cünkü üniversite olmazsa fakülte olmaz yasam döngüsü direkt biter.**

** =Compositiona örnek uml.**

** Aggregationa örnek.**

**Course’nın hocası olmasa da course objesini olusturabiliriz, yasam döngüsüne devam edebilir.**

**\*\*\*Uml’de Composition ici dolu diamond ile temsil edilir, Aggregation ise ici bos diamond ile temsil edilir.**