**Dependency Injection**

* DI, Dependency management’ın bir alt dalıdır.

**Managing Dependency**

Think about the example code where **Client** is dependent on **Service** (or **Client** has dependency on **Service**).

There may be several different scenarios regarding how a **Client** object reaches a **Service** object.



Service var ve bunu tüketen Client var.

Client’tan service’a doğru bir bağımlılık var.

**Buradaki soru şu: Client Service kısmına nasıl ulaşacak ?**

Bunun icin farklı senaryolar olabilir;

**1.Senaryo**

**Client** object creates **Service** object, which points to a highly-coupled association.

Client’ın Service objesini doğrudan olusturdugu senaryo.

Burada direkt objeyi olusturulmus ve bunu yapmamak lazım. Compositon iliskisi var gibi duruyor bu durumda high coupling ve dependencyden söz edebiliriz.



**SOLUTION**

Service objesini baska yerde olusturup client’a gecebiliriz.

* Instance variable ile gecilebilir.
* Service’I Client’ın constructruna gecebiliriz.
* Setter methoduna gecebiliriz.



Bu aynı dependency aynı olmasına rağmen composition yani ama service yaratma sorumluluğu client’ın üzerinde olmadığı icin, service ile client arasında couplingi düsürmüs ve daha sağlıklı bir noktaya getirdik diyebiliriz.



**Bu sekilde tavsiye edilmese bile instance variable sekilde, Constructora gecme seklinde ve setter methoduna gecme sekline Injection denir.**

**Yani dependency ınjection arada dependency var bunu daha rahat yönetmek adına bağımlı olunan objeyi bir baska yerde yaratıp onu kullanılacak objenin emrine vermeye Dependency Injection denir.**

**Bu anlamda Client, Service objesini yaratma sorumluluğuna sahip değildir. Client sadece ve sadece parçalarını nasıl kullanacağını bilir.**

**Bunun iyi tarafı bir şeyi kullanmak interface’ni bilmektir. Bir şeyi yaratmak ise yapısını bilmek implement kısmını bilmektir.**

**Yaratma bağımlılığından kacınmalıyız.**

**Program to an interface not an implementation.**

O yüzden ilk ihtiyacımız olan design patternlar factorylerdir.

**DI ile biz Client’ın nesne yaratma sorumluluğunu aldığımız zaman aynı durumda high-cohesion ve low-coupling duruma getirmis yani daha odaklı bir duruma getirmis oluyoruz.**