* 5 kalıp 3 ayrı gruba ayrılabilir:
* **Singleton**: Diğerlerinden farklı bir problemi çözer: Sadece tek nesne yaratma problemi.
* **Factory ve Abstract Factory**: Nesne yaratmayı soyutlar ve tekrar kullanılabilir hale getirirler.
  + Bu sayede nesne yaratma kodları, basit olsun karmaşık olsun, tek yerde olur, copy-paste ile dağılmaz.
* **Prototype ve Builder:** Nesnenin nasıl yaratılacağı ile ilgili çözüm sunarlar.
* **Prototype ve Builder** tipik olarak **Factory ve Abstract Factory** içinde kullanılırlar.
* **Prototype**, nesnenin üzerinde bulunan clone() ya da bu amaca yönelik oluşturulmuş diğer metotları kullanır,
* **Builder** ise bir süreç dahilinde nesne inşa eder.

**Dependency Injection**

* Bunlar dısında yaratımsal kalıplara **Dependency Injection (DI)** da eklenebilir.
* DI’n avantajı, tanım tabanlı (**declarative**) bir yapı sunmasıdır.
* Bu şekilde nesnelerin birbirlerine olan bağımlılıkları kodda ifade edilir ama bağımlılıkların nasıl yerine getirileceği DI altyapısı tarafından gerçekleştirilir.
* Bu da karmaşıklığı azaltır ve değişimi kolaylaştırır
* DI bir kalıp olmakla birlikte genelde teknolojik destek ile gerçekleştirilir

**Yaratımsal Kalıplar**

****

**Nerede Obje olusturalım, Her durumda obje olusturmayı saklayalım.**

**Constructor çağrısı yapmak yerine, dolayısıyla datayı istemcinin toplayıp constructora gecmesi yerine olabildigince zorunlu olanları gecip yani istemcinin bilmesi gerekenleri gecip digerlerini factory ya da abstract factory icinde pattern’in halletmesi sağlanır.**

**Yani sürekli new kullanmak yerine böyle yapıların arkasına sağlamak gerekir.**

**Nesneler arasında yakın state varsa, gecisler kolaysa prototype kullanalım.**

**Nesneler hem immutable, lazy loading, safhalı sekilde yani adım adım bina ediliyorsa builder kullanalım.**

**Bunların yanında anti-pattern olarak constructor cagrısı javaBean.**

**Bloch’un da static factory metotları da how to create them’in parçası olabilir.**

Nesne(ler)i nasıl yaratılacağına dair karar verme süreci ise şöyle modellenebilir:

****

**UYGULAMA ÖDEVİ :**

1.Aşağıda verilen problemdeki nesnelerin yaratılmalarıyla ilgili çözümleriniz neler olurdu? Modelleyip kodlayın.

• Taşımacılık yapan bir firma için yazılım tasarımı yapılacaktır.

• Kurumun kamyon, TIR, tren ve tanker olarak 4 farklı türde aracı vardır.

• Araçlara yüklenebilecek yük türleri de vardır: Kuru yük, sıvı yük, sıvı yakıt ve kuru gıda.

Kuruma, iki yer arasında taşınacak yük isteklekleri gelmekte ve siparişe dönüştürülmektedir.

• Siparişler, ön ödemeli ve sonradan ödemeli olarak iki halden birisinde oluşturulmaktadır.

• Her yeni sipariş, şöförü ve taşıma aracı ile birlikte bir projeye dönüştürülmekte ve işletilmektedir.

Projeyi gerçekleştirirken sadece nesnelerin oluşturulmalarına odaklanın, diğer iş mantıklarını modellemeye gerek yoktur.

• Nesnelerin ihtiyaç duyduğunuz alanlarını, **int** gibi olabildiğince basit tiplerden veya **String** olarak seçin.